THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à MINES ParisTech

Direct multiphase mesh generation from 3D images using anisotropic mesh adaptation and a redistancing equation

Génération de maillage à partir d'images 3D en utilisant l'adaptation de maillage anisotrope et une équation de réinitialisation

Ecole doctorale n°364

SCIENCES FONDAMENTALES ET APPLIQUEES

Spécialité MECANIQUE NUMERIQUE ET MATERIAUX

COMPOSITION DU JURY :

Mme. PEROTTO Simona
Politecnico di Milano, Rapporteur

M. REMACLE Jean-François
Université Catholique de Louvain, Rapporteur

M. MOES Nicolas
Ecole Centrale de Nantes, Président du jury

M. ORGEAS Laurent
Laboratoire 3SR, Membre du jury

M. JEULIN Dominique
MINES ParisTech, Membre du jury

M. BERNACKI Marc
MINES ParisTech, Membre du jury

M. DECENCIERE Etienne
MINES ParisTech, Membre du jury

M. COUPEZ Thierry
Ecole Centrale de Nantes, Directeur de thèse

Soutenue par JIAXIN ZHAO
le 3 Mars 2016

Dirigée par **THIERRY COUPEZ et LUISA SILVA**

Acknowledgments

Firstly, I would like to thank the director of my Ph.D., Thierry Coupez. It is a great honor to be his first Chinese Ph.D student. His ideas, knowledge, creativity and sense of humor, inspires everyone's potential.

I also thank Etienne Decencière, my co-supervisor, his patience and politeness is very impressive, and helped and guided me during my stay in Fontainebleau.

Moreover, I thank Luisa and Hugues, they take care of not only me and my work, but also my wife and my newborn baby. Their support is very special, especially for me, a new father, giving me the strength to complete my thesis and take care of my family.

I gratefully acknowledge the support from Institut Mines-Télécom and the project "De l'Image Au Maillage".

I also thank all the members of CEMEF, CMM and ICI: Marc, Rudy, Elie, François Bay, Patrick, Dominique, Catherine, Beatriz, Matthieu, François Willot, Fernand, Oliver, Patrice, Isabelle, Florence, Richard ... I want to specially thank Marie-Françoise. Since my master program, she always took care of me.

I thank all my colleagues, Jose, Abbass, Yasmina, Carole, Ghina, Jérémy, Ahmed, Nadine, Ghazza, Cyprien, Meriem, Valentine, Lucas, JF, Stephanie, Vaia, Andres, JB, Janni, Simon, Lynda, Zineb, ... and also Fuyang, Shijia, Qingxiao, Zhipeng, Yunhui and his family, Xiwei, Haisheng and Wangfang. I cannot forget my special friend Jinyuan, his words "Stay Curious for the Unknown" have always inspires me.

Finally, to my wife, my son, my parents and my family.

Abstract

Imaging techniques have well improved in the last decades. They may accurately provide numerical descriptions from 2D or 3D images, opening perspectives towards inner information, not seen otherwise, with applications in different fields, like medicine studies, material science or urban environments.

In this work, a technique to build a numerical description under the mesh format has been implemented and used in numerical simulations when coupled to finite element solvers. Firstly, mathematical morphology techniques have been introduced to handle image information, providing the specific features of interest for the simulation. The immersed image method was then proposed to interpolate the image information on a mesh. Then, an iterative anisotropic mesh adaptation operator was developed to construct the optimal mesh, based on the estimated error concerning the image interpolation. The mesh is thus directly constructed from the image information.

We have also proposed a new methodology to build a regularized phase function, corresponding to the objects we wish to distinguish from the image, using a redistancing method. Two main advantages of having such function are: the gradient of the regularized function performs better for mesh adaptation; the regularized function may be directly used for the finite element solver. Stabilized finite element flow and advection solvers were coupled to the constructed anisotropic mesh and the redistancing function, allowing its application to multiphase flow numerical simulations. All these developments have been extended in a massively parallel context.

An important objective of this work is the simplification of the image based computations, through a modified way to segment the image and by coupling all to an automatic way to construct the mesh used in the finite element simulations.

Résumé

Ces dernières années, les techniques d'imagerie ont fait l'objet de beaucoup d'améliorations. Elles permettent de fournir des images numériques 2D ou 3D précises de zones parfois invisibles l'oeil nu. Ces techniques s'appliquent dans de nombreux domaines comme l'industrie cinématographique, la photographie ou l'imagerie médicale...

Dans cette thèse, l'imagerie sera utilisée pour effectuer des simulations numériques en la couplant avec un solveur éléments finis. Nous présenterons, en premier lieu, la morphologie mathématique et la méthode d'immersion d'image. Elles permettront l'extraction d'informations permettant la transformation d'une image dans un maillage exploitable. Puis, une méthode itérative d'adaptation de maillage basée sur un estimateur d'erreur sera utilisée afin de construire un maillage optimal. Ainsi, un maillage sera construit uniquement avec les données d'une image.

Nous proposerons également une nouvelle méthodologie pour construire une fonction régulière à l'aide d'une méthode de réinitialisation de la distance signée. Deux avantages sont à noter : l'utilisation de la fonction régularisée permet une bonne adaptation de maillage. De plus, elle est directement utilisable par le solveur éléments finis. Les simulations numériques sont donc réalisées en couplant éléments finis stabilisés, adaptation de maillage anisotrope et réinitialisation.

L'objectif de cette thèse est donc de simplifier le calcul numérique à partir d'image, d'améliorer la précision numérique, la construction d'un maillage automatique et de réaliser des calculs numériques parallèles efficaces. Les applications envisagées peuvent être dans le domaine médical, de la physique des matériaux ou du design industriel.

Contents

Chapter 1

General introduction

Contents

1.1 Introduction to image-based meshing

Image acquisition is an everyday operation, in particular thanks to devices like Smartphones which have digital cameras. Taking photos allows to save memories, to capture exciting moments, ... However, classical digital cameras provide only 2D images, directly seen by our naked eyes. 3D imaging techniques exist and open new doors, offering perspectives and sometimes inner information for a better comprehension of physical phenomena. In different scientific domains, examples of 3D techniques that provide volume images are: *MRI* (Magnetic Resonance Imaging), *X-ray CT* (Computed Tomography), or nanotomography. *MRI* is based on the attenuation of energy released in different structures, by applying a gradient magnetic field detection of electromagnetic waves emitted. Then, one may detect the location and species composition of the nucleus of the object and draw an image of the internal structure. During the past few decades, research on the magnetic resonance field led six times to a Nobel Prize (Chemistry, Physics, Physiology or Medicine), illustrating the importance of this technology. *X-ray CT* scans and *X-ray* micro and nano tomography use *X-ray* to create cross-sections of a physical object which may provide a virtual model without destroying the original object. This technology has applications, for example, in both medical imaging and material science.

(a) (b) (c)

Figure 1.1: 3D techniques:(a) *MRI*-scanner;(b) *X-ray CT* scan;(c) *X-ray* Microtomography

On one hand, digital cameras, *MRI*, *X-ray*, *CT* scan, *X-ray* microtomography and other imaging techniques provide very accurate digital descriptions. On the other hand, numerical simulations based on a mesh discretization have become powerful tools in engineering designing, physical research, medical studies, ... However, an accurate geometrical representation of an object is of prime importance for this type of numerical modeling.

For all these reasons, an image-based meshing and simulation method is proposed in this work. An automatic process to create a discretization, under the mesh format, for numerical modeling and from the image data was developed. Two major techniques involved in the generation of the models are, **image processing** and **mesh generation** by adaptation of an initial coarse mesh. These techniques are also fully coupled to a finite element solver.

Image processing techniques can be used to handle the image concern, for example, image segmentation, image filtering, ... Mesh generation is also an open research field and many approaches have been used to mesh from image data, distinguished often in *CAD* (Computer Aided Design) and image based approaches. The *CAD*-based approach uses the image data to define the surface of the object

and then creates the surface elements corresponding to its boundary with traditional *CAD*-based meshing algorithms [1]. After that, the volume mesh is built from this surface mesh. On the other hand, an image-based approach combines the surface data and the mesh construction, creating the surface mesh directly. One example is the Marching Cubes algorithm [2, 3], widely used. From that, a volume mesh is created from the surface mesh with a mesh generator. In the following, a review of the techniques associated to classical image processing, like image segmentation and compression, and also related to mesh adaptation are presented.

1.2 Literature review

In this section, image characteristics and techniques on image processing are recalled. After, image compression and mesh adaptation (related to imaging) state of the art and basic notations are given.

1.2.1 Image characteristics

Here, an image is a tabular representation of some spatial phenomenon. It may be two-dimensional, such as a photograph or a screen display. With the development of imaging techniques over the last decades, three-dimensional images became very popular and used in different areas such as video games, cartoons, engineering design or medical research. There is a huge potential to find other applications in the future.

The basis of both 2D and 3D images is its digital information. It consists of a series of the smallest controllable element, the pixel for 2D images or the voxel for 3D ones. Each element contains one or several values, arranged in a particular order. Indeed, this digital information is an "image" for visualization, but is also a data array. For example, Figure 1.2 shows a popular image, "Lena" [4], which is a 8-bit grey-scale one, containing 256 different grey.

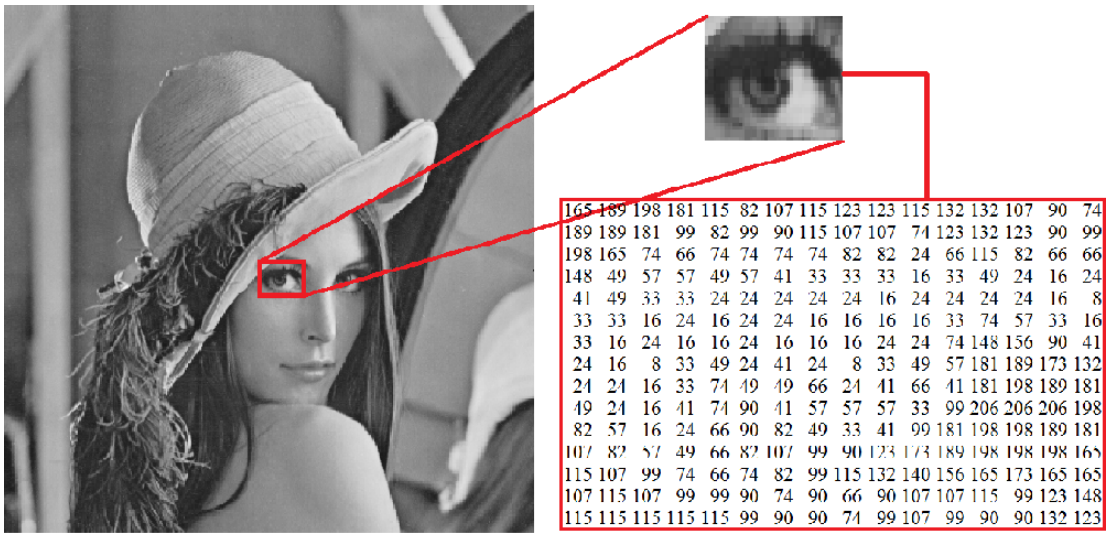


Figure 1.2: "Lena"'s image [4], and a part of it represented both as the image as it is seen and as the corresponding data array.

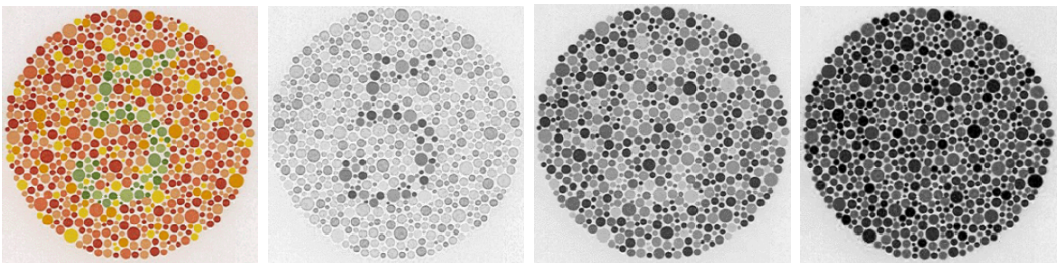
In color images, the primary colors often used are Red, Green and Blue, which correspond to trichromatic human vision. The combination of any two or three of these primary colors may create a new color. For this reason, 24-bit color images have three channels, RGB, and each channel has values within a range of $(0, 255)$, so that the number of possible colors is 16777216 ($256 \times 256 \times 256 = 2^{24}$).

To identify an object in color images, the general way to process is to decompose the image into three (RGB) channels. However, the RGB channels may not satisfy fully the demand on image processing, and is not the only way to represent the color image. Two other common representations of an RGB color model are the HSL (Hue, Saturation, Luminance) and HSV (Hue, Saturation, Value) models. Both of these two representations are widely used in computer graphics, since they are similar to the human perception of color (a strong one), but more intuitive, which may be more convenient than a RGB representation.

The Hue is used to perceive the dominant wavelength, to observe the wavelength of a pure color in a signal, to discern red, green, blue. Eyes may distinguish around 400 hues. In what concerns Saturation, it can be seen as the degree of dilution, or the inverse of the quantity of "white" in a signal. For example, a pure color is 100% saturated and may be identified as red and pink, marine blue and indigo, ... Luminance is related to the intensity of light in the signal and is used to discern the grey levels. Human eyes can perceive around 100 levels.

Thus, using different polar color spaces, with variables of intensity such as Saturation or Luminance, one may provide more options available for an appropriate representation than when using RGB channels.

As an example, let us consider a color image (Figure 1.3(a)), used to test color vision deficiencies, where people with healthy eyes can identify the number "5". This image may be decomposed into the Red-Green-Blue channels (Figures 1.3(b)(c)(d)).

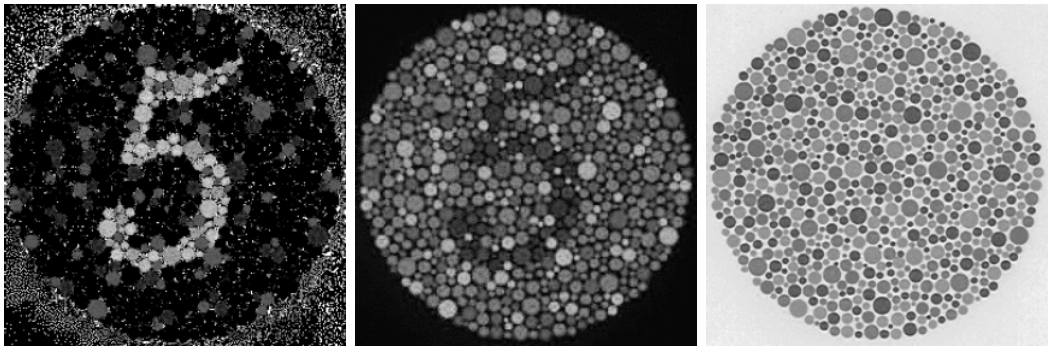


(a) Original image (b) Red channel (c) Green channel (d) Blue channel

Figure 1.3: RGB color image and the corresponding Red, Green and Blue channels.

The Red channel approximately represents the sign "5", but the remaining ones do not easily identify it. For the second option, the original image can be decomposed into its Hue, Saturation and Luminance values, shown in Figure 1.4. In the Hue one, the number "5" is very well identified and is the best between all the decomposition results.

This decomposition of a color image into different channels under several ways can then provide more options and we may choose the best one according to the objective. The usage of color image simulations will be presented in following chapters.



(a) Hue (b) Saturation (c) Luminance

Figure 1.4: HSV color image and its Hue, Saturation and Luminance values.

In computer science, image processing uses mathematical operations for handling an input image to provide an output one that preserves required characteristics, namely to satisfy visual or psychological aspects while improving other features. In addition, these techniques have a close relationship with optical theory, computer graphics and artificial intelligence and include, in particular, measurements like noise reduction, filtering, segmentation or compression.

1.2.2 Image segmentation techniques

Image segmentation is one of the fundamental tasks in image processing and computer vision. Its goal is to partition a given image into regions that contain distinct objects, with applications in medical imaging, object localization in satellite images, machine vision and others.

This section recalls the most used algorithms and techniques. In general, they may be distinguished as explicit or implicit methods. For explicit techniques, segmentation is directly based on the image data, whereas image segmentation performed with moving curves and level-set functions is often called an implicit technique.

1.2.2.1 Explicit techniques

1.2.2.1.1 Thresholding

Thresholding is the simplest method for image segmentation and is used to create, from a grey-scale image, a binary one. During the thresholding process, individual pixels/voxels in an image are marked as the object's pixels/voxels, if their value is greater than a threshold value. Typically, and according to each technique, a pixel/voxel belonging to an object is given a value of 1, while other pixels/voxels have values of 0. After that, a binary image is created by coloring each pixel/voxel is white or black, depending on the pixels/voxels' value. An example is shown in Figure 1.5, for a 2D grey-scale image, which, after application of a thresholding operation, becomes a binary image. The main drawback of this technique is that in most cases even the best possible threshold value will not lead to a convenient segmentation of the image.

Figure 1.5: Original image and resulting one, after the threshold used on this image

1.2.2.1.2 Edge detection

Edge detection is another fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature extraction, which aims at identifying points in an image at which the image intensity changes sharply or shows discontinuities [5, 6]. The purpose of edge detection is to generate an edge map based on the distribution of the intensity discontinuity of the original image. One may then define the homogeneous regions (that have the same pixel/voxel values) if the edginess measure is small enough, when compared to a given value. Classical methods of edge detection imply building an operator perceptive to large gradients in the image, and returning values of zero in uniform regions [7]. A 2D example of edge detection applied to the logo image of MINES ParisTech is shown in Figure 1.6.

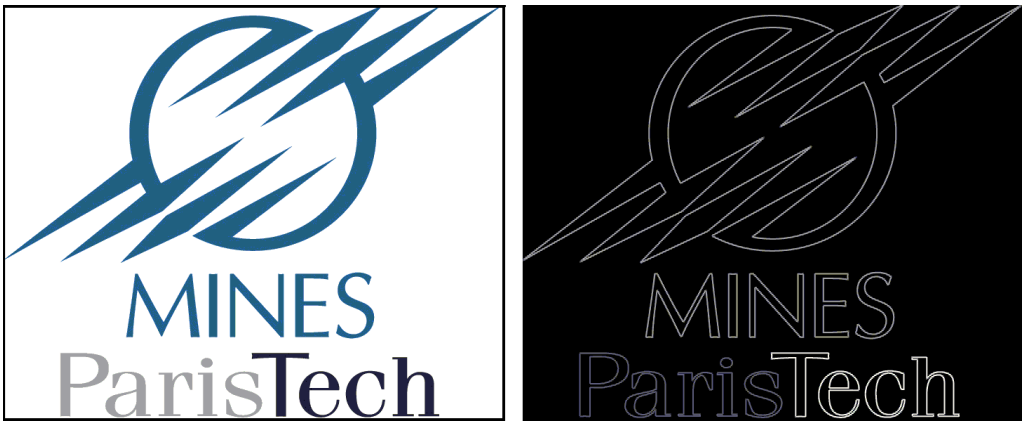


Figure 1.6: Edge detection applied on the logo image of MINES ParisTech.

However, edge detection is a difficult task in noisy images, since both the edges and noise hold high frequency content. Most usually, efforts to reduce the noise result in unclear and distorted edges. In noisy images, techniques used involve considering larger regions, to gather enough data to avoid localized noisy pixels, even if it results in a less perfect localization of the detected edges [7].

1.2.2.1.3 Region-growing methods

Region growing is a simple region-based image segmentation method, firstly proposed by [8], involving a selection of initial seed points on each region to be segmented. Regions are then iteratively grown by examining the pixel neighbors of the initial points defining the seed of a region, and by determining whether they should be added or not to the region. The difference between the pixel value or intensity and the mean value defining the region is used to allocate the pixel to the respective zone.

A suitable selection of seed points is very important, and generally depends on the user. For example, in a grey-scale image, we may examine the image histogram and choose the seed points from it, or use the connectivity or pixel adjacent information to determine the threshold and seed points. Also, by using this algorithm, noise in the image may cause a poor placement of the seeds, and is generally solved by using a mask to filter.

1.2.2.2 Implicit techniques

1.2.2.2.1 The active contour model

Active contour models were firstly proposed by Kass [9], and became very popular since: they can achieve sub-pixel accuracy of the object boundaries; various prior knowledge can be easily incorporated for robust image segmentation; the resultant contours and surfaces obtained are always are quite regular, which may be important for the use that we wish to give to the image. The fundamental idea of an active contour model is to start with a curve around the detected object, Figure 1.7. The curve moves then towards its interior normal and stops on the true boundary of the object, basing the whole procedure on an energy-minimizing model.

Let us consider a contour C, a parametrized curve, and the Mumford-Shah functional [10] restricted to the edge functional, $F(C)$, positive inside an homogeneous region, strictly zero on the boundary and negative outside. In classical active contour models, an edge (or boundary) detector is used, depending on the gradient of the initial image, $\widehat{u}$. Starting with the curve C around the object to be detected, the curve moves towards its interior normal and has to stop on the boundary of the object. For that, we compute the minimization functional as $\inf F(C)$, being

$$F(C) = \alpha \int_C |C'|^2 + \beta \int_C |C"|^2 - \lambda \int_C g(|\nabla \widehat{u}(C)|)^2 \tag{1.1}$$

where α, β and λ are positive parameters. The first two terms control the smoothness of the contour and the third term attracts the contour towards the object. In this expression, g is an edge-detection function, positive and decreasing, such that $\lim_{t\to\infty} g(t) = 0$.

1.2.2.2.2 Chan-Vese active contour model

The Chan-Vese model [11] segments the image without using the edge detection function. Let us define the evolving curve, C_0, in the original image, $\widehat{u}$, which is the

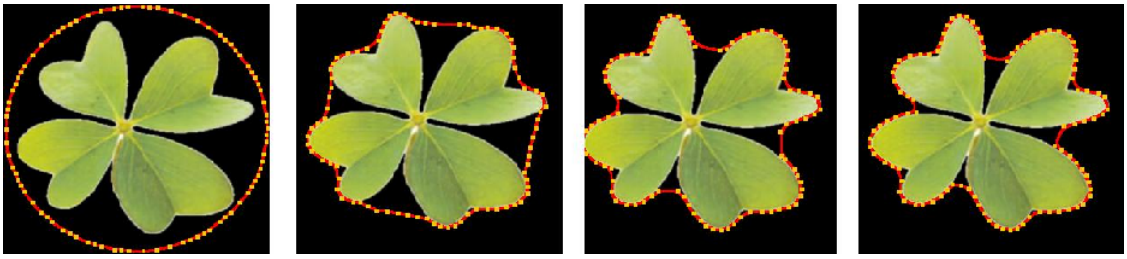


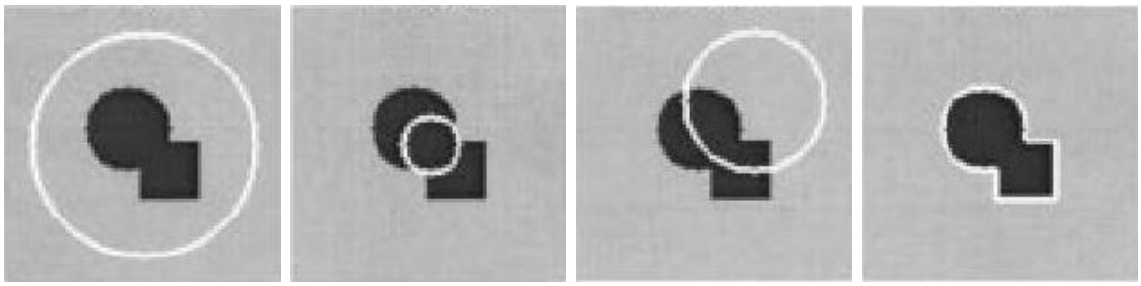
Figure 1.7: Edge or boundary detection of an object in an image using, iteratively, the classical Active Contour model.

boundary of an object. We suppose that the entire domain consists of an inside and an outside of C_0. Let us also consider the following functional, sum of two others,

$$F_1(C) + F_2(C) = \int_{inside(C)} (\widehat{u} - c_1)^2 + \int_{outside(C)} (\widehat{u} - c_2)^2 \tag{1.2}$$

where C is any other variable curve, and the constants c_1, c_2, depending on C, are the averages of $\widehat{u}$ $inside(C)$ and $outside(C)$. Thus, C_0, the boundary of the object, is the minimizer of this functional, obtained by computing $\inf_{C=C_0} (F_1(C) + F_2(C))$.

Figure 1.8 shows an example of different curves C, with the one in Figure 1.8(d) particularly minimizing the functional and detecting the boundary.



(a) (b) (c) (d)

Figure 1.8: All possible cases in the position of the curve in the Chan-Vese model: (a) the curve C is outside the object, then $F_1(C) > 0$ and $F_2(C) \approx 0$; (b) the curve C is inside the object, then $F_1(C) \approx 0$ but $F_2 > 0$; (c) the curve C is both inside and outside the object, then $F_1(C) > 0$ and $F_1(C) > 0$;(d) the functional is minimized if $C = C_0$, the curve C is on the boundary of the object [11].

1.2.2.2.3 Level-set formulation of Chan-Vese active contour model

Spline curves may be used to model the boundary of an object, but one of the most successful active contour models is the level-set method [12], presented also in [11]. In this case, the curve is represented by the zero level-set of a smooth function, which is called the level-set function. Moving the curves can be done by evolving the level-set functions instead of directly moving the curves, by solving a time-dependent PDE (particle differential equation) where the so-called velocity term reflects the image features characterizing the object to be segmented. Chan-Vese proposed a

variant of their model using a level-set function. In this case, the moving curve C is represented implicitly via a Lipschitz function u_d, such that

$$\widehat{u}_d \begin{cases} > 0 & inside(C) \in \Omega \\ = 0 & C \in \partial\Omega \\ < 0 & outside(C) \notin \Omega \end{cases} \tag{1.3}$$

Let us consider the smooth Heaviside function $H(\widehat{u}_d)$, regularized as

$$H(\widehat{u}_d) = \begin{cases} 1 & \text{if } \widehat{u}_d > \varepsilon \\ \frac{1}{2}\left(1 + \frac{\widehat{u}_d}{\varepsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\widehat{u}_d}{\varepsilon}\right)\right) & \text{if } |\widehat{u}_d| \leq \varepsilon \\ 0 & \text{if } \widehat{u}_d < -\varepsilon \end{cases} \tag{1.4}$$

The segmented image, $\widehat{u_{seg}}$, can simply be written by using the level-set formulation as:

$$\widehat{u_{seg}} = c_1 H(\widehat{u}_d) + c_2(1 - H(\widehat{u}_d)) \tag{1.5}$$

The energy functional, $F(c_1, c_2, \widehat{u}_d)$, is given by:

$$F(c_1, c_2, \widehat{u}_d) = \underbrace{\mu \int_\Omega \delta(\widehat{u}_d)|\nabla\widehat{u}_d|}_{\mu\cdot\text{Length}(C)} + \underbrace{\upsilon \int_\Omega H(\widehat{u}_d)}_{\upsilon\cdot\text{Area}(C)} + \lambda_1 \int_\Omega (\widehat{u}-c_1)^2 H(\widehat{u}_d) + \lambda_2 \int_\Omega (\widehat{u}-c_2)^2(1-H(\widehat{u}_d)) \tag{1.6}$$

where $\delta(\widehat{u}_d) = \frac{\partial H}{\partial \widehat{u}_d}$. By keeping $\widehat{u}_d$ fixed and minimizing the $F(c_1, c_2, \widehat{u}_d)$ with respect to the constants c_1 and c_2, it is easy to express these constants, as a function of $\widehat{u}_d$ by

$$\begin{cases} c_1(\widehat{u}_d) = \dfrac{\int_\Omega \widehat{u} \cdot H(\widehat{u}_d)}{\int_\Omega H(\widehat{u}_d)} = \text{average}(\widehat{u}) & \text{in } \widehat{u}_d \geq 0 \\ \\ c_2(\widehat{u}_d) = \dfrac{\int_\Omega \widehat{u} \cdot (1 - H(\widehat{u}_d))}{\int_\Omega (1 - H(\widehat{u}_d))} = \text{average}(\widehat{u}) & \text{in } \widehat{u}_d < 0 \end{cases} \tag{1.7}$$

The descent direction is parametrized using an artificial time τ, and the equation in $\widehat{u}_d(\tau)$ with the initial contour condition $\widehat{u}_d(0) = \widehat{u}_d^0$, is

$$\begin{cases} \dfrac{\partial \widehat{u}_d}{\partial \tau} = \delta(\widehat{u}_d)\left[\mu div(\dfrac{\nabla \widehat{u}_d}{|\nabla \widehat{u}_d|}) - \upsilon - \lambda_1(\widehat{u} - c_1)^2 + \lambda_2(\widehat{u} - c_2)^2\right] = 0 & \text{in } (0,\infty)\times\Omega \\ \\ \widehat{u}_d(x, y, \tau = 0) = \widehat{u}_d^0(x, y) & \text{in } \Omega \\ \\ \dfrac{\delta(\widehat{u}_d)}{|\nabla \widehat{u}_d|}\dfrac{\partial \widehat{u}_d}{\partial \vec{n}} = 0 & \text{on } \partial\Omega \end{cases} \tag{1.8}$$

where n is the outwards normal to the boundary and $\frac{\partial \widehat{u}_d}{\partial n}$ the normal derivative of $\widehat{u}_d$ at the boundary. Figure 1.9 presents the evolution of the moving curve on an original noisy image $\widehat{u}$ on the top line and the corresponding segmentation, $\widehat{u_{seg}}$, piecewise-constant, on the bottom line.

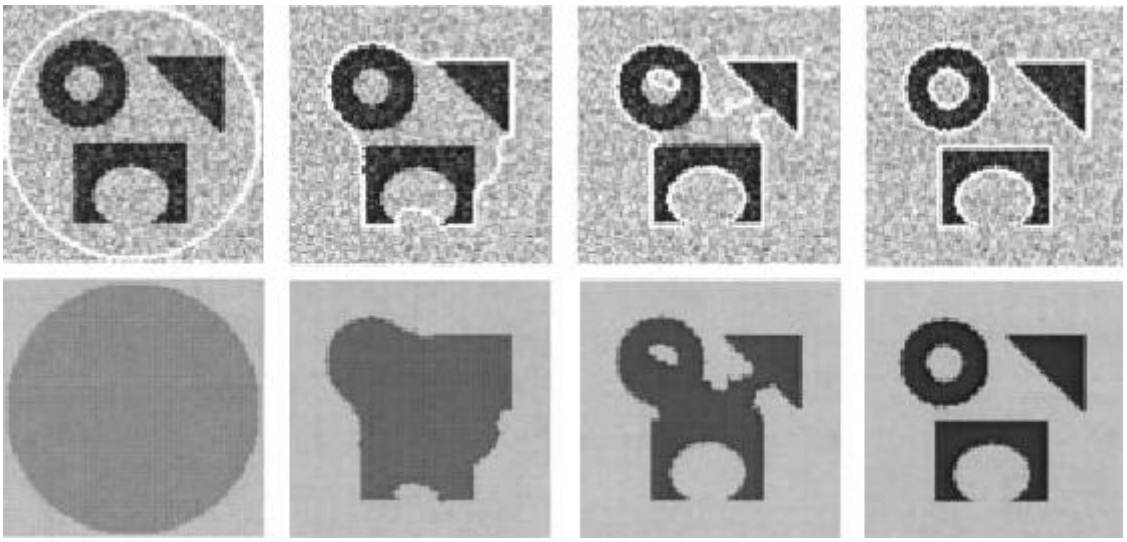


Figure 1.9: Detection of the different objects on a noisy image, with different shapes and with an interior contour. In the first line, $\widehat{u}$ and the contour, the zero of the level-set at different τ iterations; on the second line, the piecewise-constant approximation of $\widehat{u_{seg}}$ [11].

1.2.2.2.4 Multiphase Chan-Vese model

Chan and Vese extended their model to the case where we wish to segment multiple objects or phases [13]. Let us consider $m = \log_2 n$ level-set functions, $\widehat{u}_{di}$, with $i = 1, \ldots, m$, and let us consider the vector level-set function $\widehat{u}_d = (\widehat{u}_{d1}, \cdots, \widehat{u}_{dm})$, and the vector Heaviside function $H(\widehat{u}_d) = (H(\widehat{u}_{d1}), \cdots, H(\widehat{u}_{dm}))$. The union of the zero-level-sets of $\widehat{u}_{di}$ will represent the edges in the segmented image. We can define that two pixels x_1 and x_2 in Ω will belong to the same phase, if and only if $H(\widehat{u}_d(x_1)) = H(\widehat{u}_d(x_2))$. There are $n = 2^m$ possibilities for the vector values in the image of $H(\widehat{u}_d)$ or $n = 2^m$ phases, as illustrated in Figure 1.10.

Phases are labelled with I, for $1 \leq I \leq 2^m = n$, and the constant vector of averages is $c = (c_1, \ldots, c_n)$, where $c_I = \text{mean}(\widehat{u})$ corresponds to phase I, and the characteristic function for each phase I is χ_I. The Mumford-Shah functional F_n^{MS} can be written as:

$$F_n^{MS}(c, \widehat{u}_d) = \sum_{1 \leq I \leq n = 2^m} \int_\Omega (\widehat{u} - c_I)^2 \chi_I + \mu \sum_{1 \leq i \leq m} \int_\Omega |\nabla H(\widehat{u}_{di})| \tag{1.9}$$

leading to an evolution equation per level-set of the following type: given $\widehat{u}_{di}(0) = \widehat{u}_{di}$

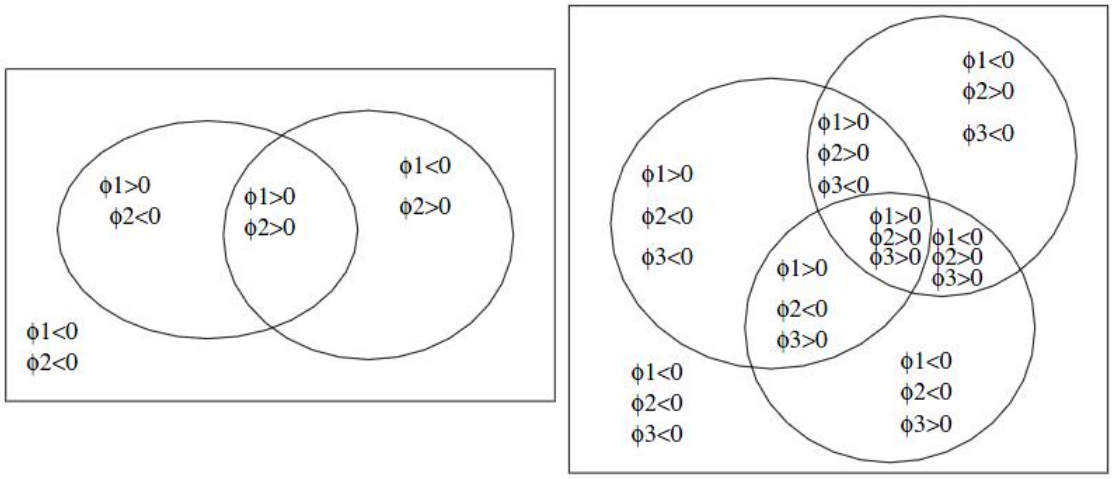


Figure 1.10: On the left, the two curves $\{\widehat{u}_{d1} = 0\} \cup \{\widehat{u}_{d2} = 0\}$ that partition the domain into 4 regions: $\{\widehat{u}_{d1} > 0, \widehat{u}_{d2} > 0\}$, $\{\widehat{u}_{d1} > 0, \widehat{u}_{d2} < 0\}$, $\{\widehat{u}_{d1} < 0, \widehat{u}_{d2} > 0\}$, $\{\widehat{u}_{d1} < 0, \widehat{u}_{d2} < 0\}$. On the right, three curves $\{\widehat{u}_{d1} = 0\} \cup \{\widehat{u}_{d2} = 0\} \cup \{\widehat{u}_{d3} = 0\}$ partition the domain into 8 regions: $\{\widehat{u}_{d1} > 0, \widehat{u}_{d2} > 0, \widehat{u}_{d3} > 0\}$, $\{\widehat{u}_{d1} > 0, \widehat{u}_{d2} > 0, \widehat{u}_{d3} < 0\}$, $\{\widehat{u}_{d1} > 0, \widehat{u}_{d2} < 0, \widehat{u}_{d3} > 0\}$, $\{\widehat{u}_{d1} > 0, \widehat{u}_{d2} < 0, \widehat{u}_{d3} < 0\}$, $\{\widehat{u}_{d1} < 0, \widehat{u}_{d2} > 0, \widehat{u}_{d3} > 0\}$, $\{\widehat{u}_{d1} < 0, \widehat{u}_{d2} > 0, \widehat{u}_{d3} < 0\}$, $\{\widehat{u}_{d1} < 0, \widehat{u}_{d2} < 0, \widehat{u}_{d3} > 0\}$, $\{\widehat{u}_{d1} < 0, \widehat{u}_{d2} < 0, \widehat{u}_{d3} < 0\}$[13]. In the image, ϕ is the level-set function, $\widehat{u}_d$.

$$\frac{\partial \widehat{u}_{di}}{\partial \tau} = \delta(\widehat{u}_{di})\mu div(\frac{\nabla \widehat{u}_{di}}{|\nabla \widehat{u}_{di}|}) - \sum_{1 \leq I \leq n = 2^m} (\widehat{u} - c_I)^2 \chi_I \tag{1.10}$$

Figure 1.11 illustrates this with an example of [13], using two curves to segment four-phases on a noisy synthetic image.

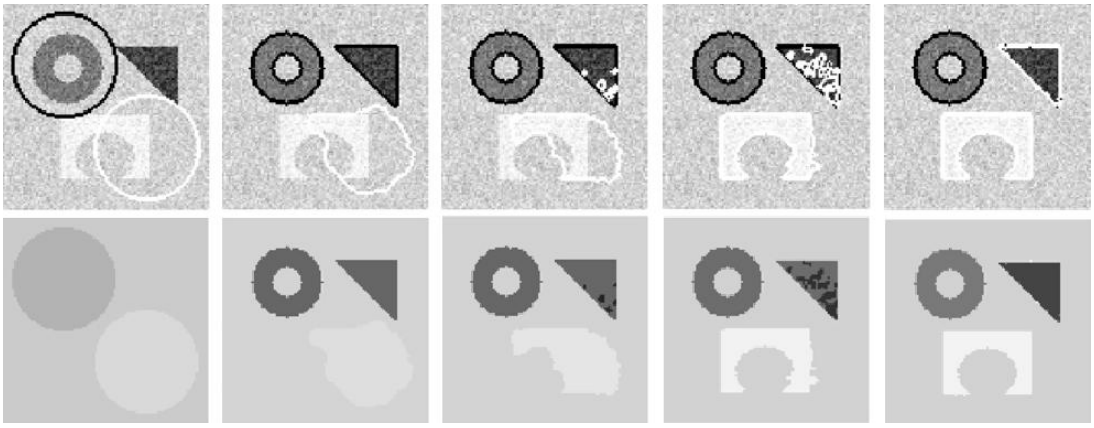


Figure 1.11: Segmentation of a noisy synthetic image, using the 4-phase piecewise constant mode. The first line represents the evolving contours overlay on the original image and the second line gives the computed averages of the four segments c_{11}, c_{10}, c_{01}, c_{00} [13].

The authors proposed also an extension to multichannel-multiphase segmentation for color RGB images, $\widehat{u} = (\widehat{u_1}, ..., \widehat{u_N})$, with $N = 3$ channels. For each channel

$i = 1, \ldots, n$, we have the constants $c_I = (c_{I,1}, \ldots, c_{I,N})$ and the built functional is

$$F_n(c_I, \widehat{u}_d) = \sum_{1 \le I \le n = 2^m} \sum_{i=1}^{N} \int_\Omega (\widehat{u}_i - c_{I,i})^2 \chi_I + \sum_{1 \le I \le n = 2^m} \mu \int_\Omega |\nabla H(\widehat{u}_{di})| \qquad (1.11)$$

1.2.3 Image compression

The purpose of image compression is to reduce the redundant information to have more efficient data storage and transmission formats. Image compression can be distinguished in lossy compression and lossless compression. When lossless compression is used then, after compression, the information is not lost and may be fully restored to its original status. On the other hand, when little perceptible loss is acceptable (and sometimes imperceptible) but also allows an approximate reconstruction of the original data, one has lossy compression, greatly reducing the bit rate and achieving a better compression ratio. In a high bit rate image compressed into a low bit rate, the definition of the compression ratio is:

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}} \qquad (1.12)$$

1.2.3.1 Lossy compression

In this method, the color palette index representing the selected color is defined in the image header compression palette firstly. Then, each pixel references the index, which may reduce the common color space in the image.

An extension, the chroma subsampling method, takes advantage of the human eye sensitivity to changes in brightness much larger than the color change, so that the color image information can be reduced by half or more [14].

Transform coding is the most commonly used method [15], employing Fourier-related transforms, such as the DCT (Discrete Cosine Transform) or quantized and entropy coding compression methods like $JPEG$ (Joint Photographic Experts Group) compression [16]. This may achieve $10:1$ compression ratios with little loss in the image quality.

Fractal compression was firstly realized in [17], by compressing the image to suit for textures and natural images. Some pictures seem very complicated for human eyes, but which may only contain capture a very low amount of information. Fractal algorithms convert these data through "fractal coding", and compress common features by self-similarity compression.

1.2.3.2 Lossless compression

Run-length encoding is the simplest way for data compression. It can be explained with a simple example: the data list "AAAAABBBBCCCDDE", 15 characters, can be replaced by the "5A4B3C2DE" one with 9 characters. This method is well suited for already images, but not for continuous ones. It has been applied to the BMP (bitmap) or $TIFF$ (Tagged image file format) image formats [18].

Other methods concern adaptive dictionary algorithms, such as the Lempel-Ziv-Welch one, proposed in [19, 20] and used for GIF (Graphics Interchange Format) image formats, with a fixed length code editor to store different lengths of strings. The advantage is that requires a small storage table and is highly efficient, without any data analysis.

1.2.4 Mesh adaptation

Over the last decades, numerical simulation played a very import role in the industrial development. It has reduced the experimental costs and provided a solution for design, avoiding potential risks. The basic idea of numerical simulation is to replace the continuous domain into a set of discrete sub-domains (elements of a mesh, for example), couple it to a numerical solver (Navier-Stokes, thermal solver, ...), to have an approximate solution in the discretization domain.

However, considering the computational cost and time to have very accurate solutions, development of optimal meshing techniques is a field undergoing continuous improvement. Mesh adaptation techniques are thus efficient solutions, providing dynamic meshes adapted to the solution fields, based on the interpolation error estimation.

1.2.4.1 Topological optimization mesh generator

Let us first recall some classically mesh generation algorithms and go more deeply on the description of the topological optimization mesh generator, used in this thesis.

For mesh generation, the Delaunay triangulation $\mathcal{D}$ algorithm is a very popular one[21]. Let us consider a set of points $\mathbf{X}$ in the Euclidean space $\mathbb{R}$, so that a Delaunay triangulation is noted as $\mathcal{D}(\mathbf{X})$, for $\mathbf{X}$ inside the circumcircle of each triangle in $\mathcal{D}(\mathbf{X})$. This method [22, 23] then creates elements from the boundary of domain. The mesh is then refined inside of the domain by inserting nodes, but always maintaining the Delaunay triangulation property. Extension with a Quadtree-Octree algorithm has been proposed [24], and applied in many commercial numerical simulation softwares. This algorithm has also been widely used in digital image processing [25], and elements of the mesh may not only be triangle/tetrahedron, but also rectangles/cubes, providing more options than classical Delaunay meshing. There are also some other mesh techniques like, for example, the advancing front method [26].

In this work, a topological optimization mesh generator is used, developed by Coupez [27, 28, 29]. It is based on the iterative improvement of an initial unsatisfactory mesh by performing local operations. Before detailing the methodology, let us present some notations, properties and definitions on mesh topologies.

Let us note the mesh, $\mathcal{H}$, of a computational domain $\Omega \subset \mathbb{R}^d$. K is an element of this mesh, being a d-simplex with the convex hull of $d+1$ vertices, and $\mathcal{K}$ is a the set of elements. For example, a $1D$-segment has 2 vertices, a $2D$-triangle has 3 vertices and the $3D$-tetrahedron has 4 vertices. Let $\mathcal{F}(\mathcal{K})$ be the set of faces of $\mathcal{K}$'s elements, being a face designated by F. A basic property is

$$1 \leq \text{card}(\mathcal{K}(F)) \leq 2, \ \forall F \in \mathcal{F} \tag{1.13}$$

and the faces belonging to the boundary of $\mathcal{K}$ are:

$$\partial\mathcal{K} = \{F \in \mathcal{F},\ card(\mathcal{K}(F)) = 1\} \tag{1.14}$$

Each face $F \in \mathcal{F}(\mathbf{K})$ shares no more than two elements, and only the boundary faces have one element.

The operation of local modification of the mesh topology is done by replacing the chosen sub-set $\mathcal{K}_a$ by another sub-set $\mathcal{K}_b$, where $\mathcal{K}_a \subset \mathcal{K}$ and $\partial\mathcal{K}_a = \partial\mathcal{K}_b$. In other words, the new sub-set $\mathcal{K}_b$ has the same boundary as the replaced sub-set $\mathcal{K}_a$, and a node $\mathbf{X}_a$ connects to the boundary faces that do not contain it. Let us denote the set of nodes and faces of $\mathcal{K}_a$ by $\mathcal{N}(\mathcal{K}_a)$ and $\mathcal{F}(\mathcal{K}_a)$, in the mesh topology $\mathcal{K}_a$, and by $\mathcal{C}(\mathcal{K}_a)$ the centroid of the nodes of $\partial\mathcal{K}_a$. A cut/paste operation, designated θ, is performed. Let $\mathbf{X}_a \in \mathcal{N}(\mathcal{K}_a) \cup \mathcal{C}(\mathcal{K}_a)$ then the new optimal mesh topology, $\widetilde{\mathcal{K}}$, is given by (Figure 1.12):

$$\widetilde{\mathcal{K}} = \theta(\mathcal{K}) = \mathcal{K} - \mathcal{K}_a + \underbrace{\mathcal{K}_b}_{\mathcal{K}(\mathbf{X}_a, \partial\mathcal{K}_a)} \tag{1.15}$$

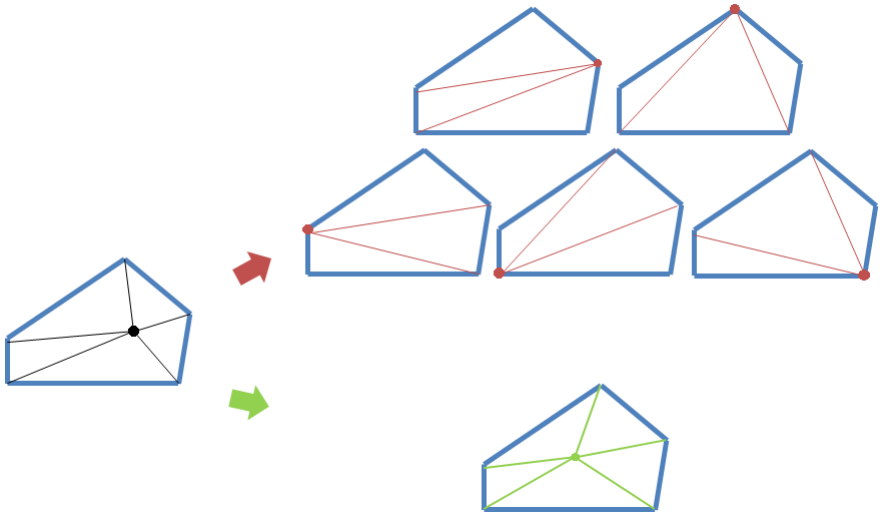


Figure 1.12: Example of sub-set $\mathcal{K}_a$ replaced by $\mathcal{K}_b$, by adding the new nodes with the boundary or around the centroid $\mathcal{C}$.

The algorithm can only handle the nodes inside of domain, but not the nodes of the boundary. To overcome this drawback, the author [28] proposed to insert a virtual node 0, which connects to all the nodes along the boundary (as illustrated in Figure 1.13). Then, each node becomes an "inside" one and can be handled by the presented local modification operation. More details are given in [27, 28, 29]. Indeed, the operation θ may iteratively modify the local mesh, until no significant improvement is necessary.

1.2.4.2 Criteria of optimal local mesh topology

The previous section has presented the based idea of optimization of local mesh topology. In this process, two criteria are enforced in a compulsory way:

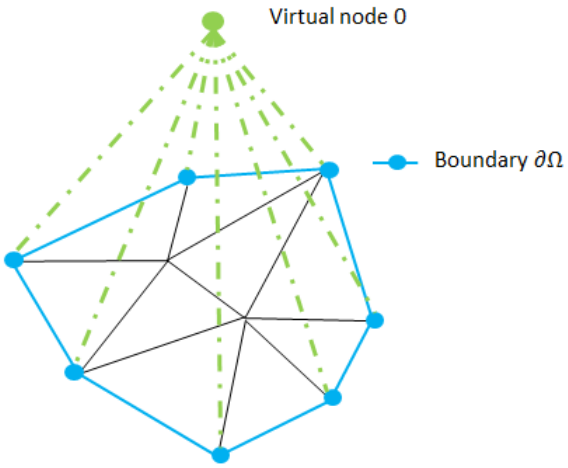


Figure 1.13: Virtual node connected to boundary nodes, generating the virtual elements.

- the **minimal volume** principle, which ensures the conformity of mesh, with no element overlaps. If $\mathcal{K}$ denotes the set of elements filling the domain Ω, we choose an optimal sub-set $\mathcal{K}_b$ which improves $\mathcal{K}_a$ and is given by:

$$\mathcal{K}_b = \arg\min_{\mathcal{K}} \sum_{K \subset \mathcal{K}} |\text{Volume}(K)| \tag{1.16}$$

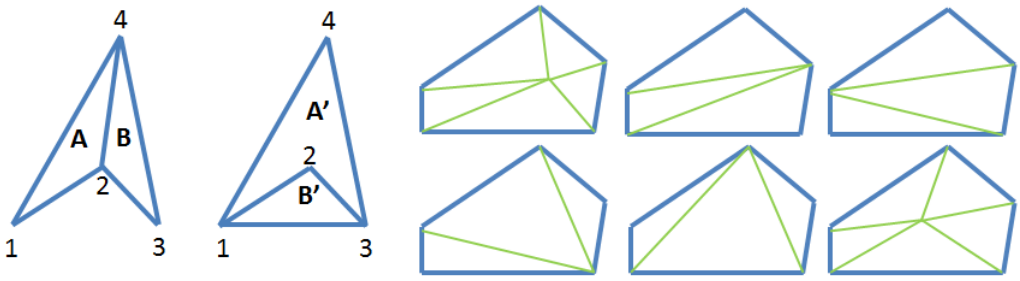
where $|\text{Volume}(K)|$ means the volume of the element K, $|\text{Volume}(K)| = \int_K dK$. As shown in Figure 1.14(a), four nodes may construct two possible mesh topologies. The left one satisfies the minimal volume criteria, but not the right one since:

$$|\text{Volume}(K_{A'})|+|\text{Volume}(K_{B'})| > |\int_{\substack{1 \rightarrow 2 \\ \uparrow \quad \downarrow \\ 4 \leftarrow 3}} dK| = |\text{Volume}(K_A)|+|\text{Volume}(K_B)| \tag{1.17}$$

However, the mesh topology satisfies the minimal volume property, which may not be unique, as illustrated in Figure 1.14(b). The following criteria may overcome this problem, and choose the best mesh topology.

- the **geometrical quality** principle, which picks the one with the highest geometrical quality of the elements among all possible triangulations, which involved shift, destruction and creation of nodes. The evaluation of the quality of each element of the mesh topologies is done by computing:

$$Q(K) = \frac{|\text{Volume}(K)|}{h_K^d} \tag{1.18}$$



(a) (b)

Figure 1.14: Criteria of optimal local mesh: (a) minimal volume criteria; (b) possible triangulations of elements for the same minimal volume.

where h is the space dimension and h_K represents the mean of the edge lengths. The geometrical quality varies between 0 and 1, with 1 being the best quality.

1.2.4.3 Mesh generation algorithm

The definition of mesh optimization and the criteria have been well presented are summarized in the following algorithm. Firstly, we suppose an initial mesh of Ω, with a connected boundary, $\partial\Omega$. Then, the recursive local mesh optimization algorithm is as follows:

Algorithm 1: Algorithm of local mesh topology optimization

Input: $(\mathcal{N}, \partial\mathcal{K})$, a mesh topology of domain Ω
Output: $(\mathcal{N}_{opt}, \partial\mathcal{K}_{opt})$, the optimal mesh topology
1 **while** *the mesh topology* $\mathcal{K}$ *has to be improved* **do**
2 **for** *each node and edge of the mesh topology* **do**
3 Cut the local topology $\mathcal{K}_a$, associated with nodes or edges.
4 Paste it to a new local mesh topology, $\mathcal{K}_b = \mathcal{K}(\mathbf{X}_a, \partial\mathcal{K}_a)$, which minimizes the volume and maximizes the element's qualities among all the candidates, to obtain $(\mathcal{N}_{opt}, \partial\mathcal{K}_{opt})$

1.2.4.4 H-refinement method

In this section, the adaptive mesh refinement method will be presented in a general way. It provides a dynamic mesh during the simulation, adapted to the computed solution. The "sensitive" sub-domain targeted is always indicated with an interpolation error estimator, related to the high solution gradient areas.

The error is given by the difference between the exact solution and the one computed on the discretized domain. However, we do not know the exact solution, and the error may be the result of physical or mathematical errors, of the discretization (mesh size), of data inputs,... Different factors may affect this error and *a priori* error estimators have been introduced. They estimate the size of a solution or its

derivatives of a partial differential equation [30] and also provide the discretization error in a current mesh, instead of the actual error.

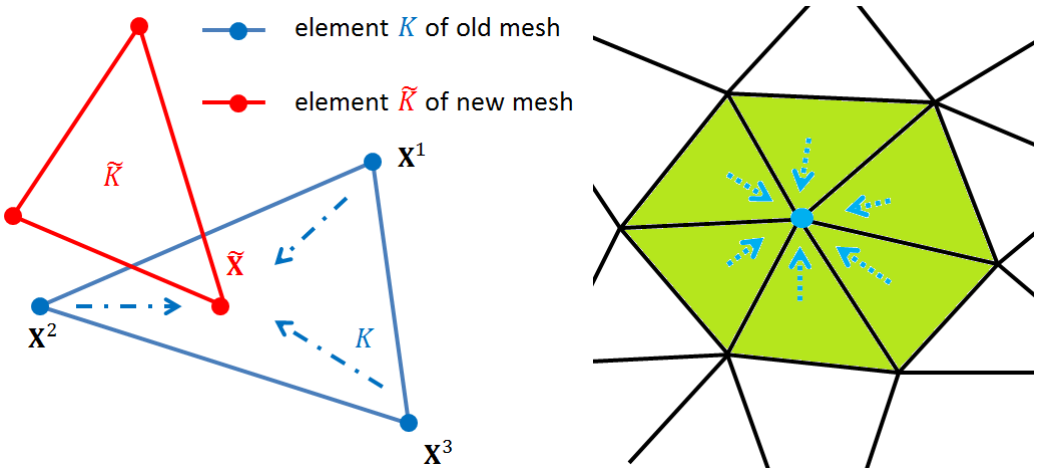
The H-refinement method is the one applied to modify the topology of mesh in this work. Based on the current mesh, the interpolation error estimator indicates the error estimate over the whole computation domain. The presented mesh generator may improve the mesh size by inserting or removing nodes to reduce or balance this error estimate. The way of computing this estimate will be described in Chapter 2.

1.2.4.5 Transfer of data between meshes

Let us suppose that mesh adaptation has been done, the new improved mesh replacing the old one. The data transfer from the old mesh to the new one without loosing information is of prime important. In our case, two types of information are stored within the mesh, such as nodal or element variables and parameters, allowing the construction of the field through **P1** or **P0** interpolations.

To transfer **P1** data, a method called "P1 to P1" is used. Figure 1.15 illustrates the element K of the old mesh and the element $\widetilde{K}$ of the new mesh, as well as element node and its value denoted, respectively, as $\mathbf{X}$ and and $u(\mathbf{X})$. Firstly, we identify the new node $\widetilde{\mathbf{X}}$ inside the old element K and its barycentric coordinate in K, named as $coord(\widetilde{\mathbf{X}})_K$. The value of node $\widetilde{\mathbf{X}}$ is computed using a linear interpolation from the nodal values of K and $coord(\widetilde{\mathbf{X}})_K$, as

$$u(\widetilde{\mathbf{X}}) = \sum_{i=1}^{d+1} coord(\widetilde{\mathbf{X}})_K \cdot u(\mathbf{X}^i) \tag{1.19}$$



(a) (b)

Figure 1.15: (a) Transfer of P1 data: "P1 to P1" procedure; (b) Transfer of P0 data to P1, with a Superconvergent Patch Recovery method.

To transfer **P0** data, the method is also called "P0-P1-P0". Using a Superconvergent Patch Recovery procedure [31], the values at the elements are distributed on the nodes, as seen in Figure 1.15. Then, applying the "P1 to P1" procedure, data at all the nodes is obtained for the new mesh. Finally, the Gaussian integral rule may compute the value at the element through its $d+1$ nodal data.

1.2.5 Finite element method

Computational modeling starts with a representation of the physical phenomena using a mathematical model and the governing equations of the problem to be solved. The equations to be solved are often described by $PDEs$ (partial differential equations) and the numerical solutions of these mathematical models approximate the exact solutions. In general, three classical methods are widely used and developed to solve the $PDEs$: finite difference method, finite volume method and finite element methods.

The finite difference method is based on Taylor series to approximate the $PDEs$. However, this method has a limit on handling complex geometries, since several layers of discretization nodes are asked, increasing the computational cost.

The finite volume method is used to integrate over a cell (volume), respecting the conservation laws and assuming piecewise constant approximation spaces. The values of the conserved variables are located within the volume element, and not at the nodes or surfaces, which are calculated using the mesh geometry. It has been applied in many commercial simulation softwares.

In this work, we have used the finite element method to solve the $PDEs$. Based on a Galerkin formulation, it provides the solutions of many small sub-domains by integral forms that approximate the equations over the entire domain. Furthermore, to avoid oscillations with convective-dominated terms and stabilize mixed-finite element methods, specific stabilization methods are also applied, such as the streamline upwind Petrov-Galerkin method, proposed by [32, 33] and the Residual Free Bubble approach [34, 35] or the Variational MultiScale method [36].

1.3 Objective of the thesis

In this thesis, we propose a new technique for image-based meshing, an alternative and innovative way to skip surface mesh construction by directly using image data. Then, volume mesh is built with an automatic topological optimization mesh generator, coupled to an estimation error procedure. Additionally, mathematical morphology methods are used to handle image data, to ensure that they are suitable for mesh adaptation and numerical simulation. On the other hand, a newly proposed redistancing level-set approach is implemented to rebuild a phase function for object to be identified in an image, a "smooth" segmentation, and the rebuilt level-set function will be used in numerical simulations. Finally, flow computations on the images are performed, with a dynamic coupling between image-mesh-flow solver.

1.4 Framework of the thesis

This thesis work is a part of the project "De l'Image au Maillage", supported by Institut Mines-Télécom, a collaboration between several researchers from five Écoles des Mines (Paris, Saint-Etienne, Albi-Carmaux, Douai, Alès). The first part of this work was performed at the Center for Material Forming, "CEntre de Mise En Forme des matriau" (CEMEF) at Sophia Antipolis, to handle the mapping of the

image data to the mesh structure. The second part was achieved at the Center for Mathematical Morphology, "Centre de Morphologie Mathématique" (CMM), at Fontainebleau, to interface our implementations with Morph-M, a powerful tool for image processing, based on mathematical morphology techniques. Finally, at the Institute of High Performance Computing, "Institut de Calcul Intensif" (ICI) of École Centrale de Nantes, implementations have been extended to a massively parallel context, using the ICITech library and numerical simulations based on real images have been performed.

1.5 Layout of the thesis

This thesis is divided into six chapters. Chapter 1 is the general introduction of this work. Chapter 2 introduces a new methodology to create anisotropic meshes based on image data. Anisotropic mesh adaptation is constructed using metric tensors, which are computed from the interpolation error estimate of the image data on the mesh. The interpolation of the data on the mesh and its adaptation have also been parallelized. Chapter 3 gives a new methodology to build a continuous phase function per object of a segmented image, by a redistancing procedure, coupled to mesh adaptation. Image processing techniques were implemented to improve and accelerate this redistancing-adaptation procedure. Chapter 4 describes the numerical approach used for multiphase flow problems, the Variational Multiscale Navier Stokes solver, based on a stabilized finite element method. Then, numerical simulations on real images are illustrated, including fluid-structure interactions or object's interfaces dynamics. Finally, conclusions and perspectives are presented.

1.6 Résumé en français

Ce chapitre introduit les travaux réalisés dans cette thèse. Premièrement, nous rappelons les techniques de traitement d'image, comme la segmentation ou la compression. Deuxièmement, une brève étude des différentes méthodes de géneration et d'adaptation sont présentées, avec un intêret particulier pour les techniques avec optimisation de topologie de maillage. Ensuite, nous décrivons les objectifs poursuivis, avec la réalisation de simulations numériques par la méthode des éléments finis et directement sur des images. La morphologie mathématique est utilisée pour traiter les données de l'image, afin de s'assurer qu'ils sont mieux exploitables par les outils d'adaptation de maillage et la simulation numérique. D'autre part, une méthode de reinitialisation des fonctions de phase est proposée pour reconstruire une fonction continue par objet à identifier dans une image, nécessaire aux simulations d'écoulements multiphasiques.

Chapter 2

The immersed image method

Contents

2.1 Introduction

In this section, the Immersed Image Method is introduced, consisting mainly in the interpolation of an image onto a given mesh. Coupled to anisotropic adaptation, it allows a very good representation of the original image, with fewer number of nodes than the number of pixels/voxels and creates simultaneously an adapted anisotropic mesh. The previous Chapter has presented the definition behind an image and a mesh. The main idea of the Immersed Image Method is to map the pixels (2D)/voxels (3D) value of the image onto an existing 2D/3D mesh, allowing us to build a function within it. In classical methods, from this function we recognize N phases by building $N-1$ phase functions, being $N-1$ surface meshes then extracted. In image-based approaches for mesh generation, the most used method to create a surface mesh is the Marching Cubes [2, 3], building after the volume one. Here, we propose an alternative way, by skipping these two steps, and directly construct the mesh. This is specially interesting in the case where computations will be performed on the adapted mesh, constituted of all the phases. For that, we propose to use directly the image data and to minimize the *a posteriori* immersed image interpolation error by adapting anisotropically the mesh [27, 28, 37, 38].

2.2 Interpolation of the image Pixel/Voxel values on the mesh

A new method to transform the image information in a mesh is presented in this section. In general, a pixel/voxel is the smallest controllable surface/volume element of a two/three dimensional image, arranged in this latter in a specific order. The image, noted as $\widehat{u}$, can be in two ($\widehat{L} \times \widehat{H}$) or three dimensions ($\widehat{L} \times \widehat{H} \times \widehat{W}$), with $\widehat{T}$ pixels/voxels ($\widehat{T} = \widehat{L} \times \widehat{H}$ or $\widehat{T} = \widehat{L} \times \widehat{H} \times \widehat{W}$), where $\widehat{L}$, $\widehat{H}$, $\widehat{W}$ are the length, height and width of the image. Therefore, the set of pixels/voxels in the image $\widehat{u}$ is defined as:

$$\begin{aligned} Pixel &= \{Pixel^k \in \mathbb{R}^2,\ \forall k = 1, \dots, \widehat{T}\} \\ Voxel &= \{Voxel^k \in \mathbb{R}^3,\ \forall k = 1, \dots, \widehat{T}\} \end{aligned} \tag{2.1}$$

Let $\widehat{U^k}$ be the value and $(\widehat{l^k}, \widehat{h^k})/(\widehat{l^k}, \widehat{h^k}, \widehat{w^k})$ the coordinates of $\widehat{u}$ at $Pixel^k/Voxel^k$, such that: $\widehat{u}(Pixel^k) = \widehat{U^k}$ or $\widehat{u}(Voxel^k) = \widehat{U^k}$, $\forall\ \widehat{l^k} = 1, \dots, \widehat{L}$, $\widehat{h^k} = 1, \dots, \widehat{L}$ and $\widehat{w^k} = 1, \dots, \widehat{W}$.

$$\widehat{u} = \begin{cases} \sum\limits_{i}^{\widehat{T}} \widehat{u}(Pixel^k) = \sum\limits_{i}^{\widehat{T}} \widehat{U^k} \\ \sum\limits_{i}^{\widehat{T}} \widehat{u}(Voxel^k) = \sum\limits_{i}^{\widehat{T}} \widehat{U^k} \end{cases} \quad \forall\ k = 1, \dots, \widehat{T} \tag{2.2}$$

The relation between the k-th pixel/voxel and its coordinate $(\widehat{l^k}, \widehat{h^k}, \widehat{w^k})$ is:

$$k = \widehat{l^k} + \widehat{h^k} * \widehat{L} \underbrace{+\widehat{w^k} * \widehat{H} * \widehat{L}}_{\text{only for voxel}} \tag{2.3}$$

Like referred previously, for a 8-bit grey-scale image, there are only $2^8 = 256$ possible levels and the value of $\widehat{U^k}$ varies between 0 (black color) and 255 (white color). However, for color images, like a RGB (Red, Green, Blue), a three channels are usually used, each channel with 256 levels, so that there are $2^{24} = 16777216$ possible colors.

Let us recall the definition of a 2D/3D mesh, which is a collection of nodes, edges and elements (for example, triangular or quadrilateral for a 2D mesh, tetrahedron or pyramidal for a 3D mesh). The mesh support is widely used to discretize geometries in finite element or finite volume methods. For an initial 2D/3D mesh of a domain of size $([0, \mathrm{X}] \times [0, \mathrm{Y}])$ or $([0, \mathrm{X}] \times [0, \mathrm{Y}] \times [0, \mathrm{Z}])$ with N nodes, we define:

$$\mathcal{N} = \{\mathbf{X}^i \in \mathbb{R}^d, \forall i = 1, \ldots, N\} \tag{2.4}$$

as the set of nodes of the mesh and d its spatial dimension. We denote $U^i = u(\mathbf{X}^i)$ as the nodal value of u at node $\mathbf{X}^i$, with the coordinate of node $\mathbf{X}^i$ being (x^i, y^i) or (x^i, y^i, z^i) for a 2D or 3D mesh, with $x^i \in [0, \mathrm{X}], y^i \in [0, \mathrm{Y}], z^i \in [0, \mathrm{Z}]$.

During image to mesh processing, the image, under one of different possible formats, is firstly read and associated to a data array. Then, this data array of pixel/voxel values from the image $\widehat{u}$ is interpolated into the mesh, providing a distributed field, named here u. The interpolation equations are as follows, obtained using the coordinates of the pixel/voxel $(\widehat{l^k}, \widehat{h^k}, \widehat{w^k})$ and of the nodes (x^i, y^i, z^i):

$$\begin{aligned} &u(\mathbf{X}^i) = \widehat{u}(Pixel^k/Voxel^k), \text{ where } \widehat{l^k} = \mathrm{int}(\tfrac{x^i}{\mathrm{X}} \cdot (\widehat{L} - 1) + 1), \\ &\widehat{h^k} = \mathrm{int}(\tfrac{y^i}{\mathrm{Y}} \cdot (\widehat{H} - 1) + 1), \underbrace{\widehat{w^k} = \mathrm{int}(\frac{z^i}{\mathrm{Z}} \cdot (\widehat{W} - 1) + 1)}_{\text{only in the 3D case}} \ \forall\ i = 1, \ldots, N \end{aligned} \tag{2.5}$$

In fact, the solution $u \in \mathbb{C}^2(\Omega) = \mathcal{V}$. The discretized functional space corresponding to $\mathcal{V}$, $\mathcal{V}_h$, is a simple P^1 finite element approximation space with a piecewise constant field, such that:

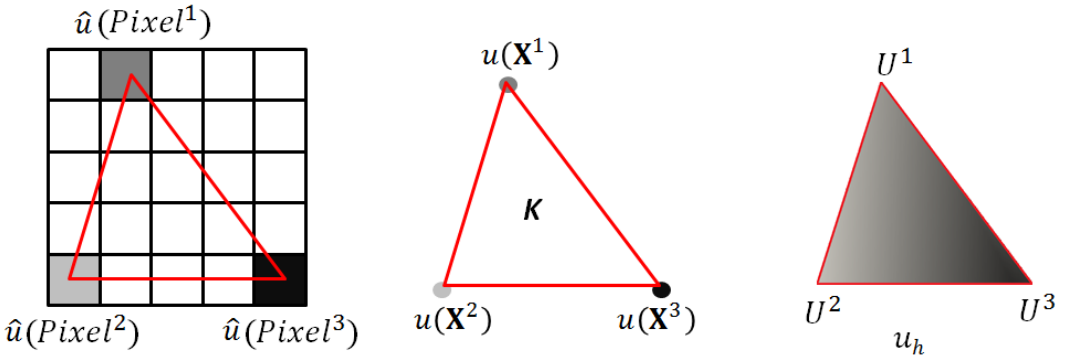
$$\mathcal{V}_h = \left\{u_h \in \mathbb{C}^0(\Omega), u_h|_K \in P^1(K), K \in \mathcal{K}\right\} \tag{2.6}$$

where $\Omega = \bigcup_{K \in \mathcal{K}} K$ is the computational domain, K is the element, in our case a simplex (segment, triangle, tetrahedron) and $\mathcal{K}$ is the set of elements of the mesh, $\mathcal{H}$. Let Π_h be the Lagrange interpolation operator from $\mathcal{V}$ to $\mathcal{V}_h$, so that:

$$\Pi_h u(\mathbf{X}^i) = u(\mathbf{X}^i) = U^i, \ \forall\ i, \ldots, N \tag{2.7}$$

More generally, we will designate the continuous form represented as the interpolation solution, u_h. Figure 2.1 presents a simple interpolation example, of an image $\widehat{u}$ with 25 pixels ($\widehat{L} \times \widehat{H} = 5 \times 5$) on a single triangle element K, with 3 nodes $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3$. As given by Equation (2.5), the three nodes can directly be mapped

on the image and will have the values of the pixels $\widehat{u}(Pixel^1)$, $\widehat{u}(Pixel^2)$, $\widehat{u}(Pixel^3)$, as illustrated in Figures 2.1(a)(b). Since the values of the three pixels have been interpolated on the mesh nodes, the Lagrange interpolation solution on element K is computed with the operator $\Pi_h u(\mathbf{X}^i)$ from the three nodal values, as shown in Figure 2.1(c).



(a) Image map (b) $u(\mathbf{X}^i)$ (c) u_h

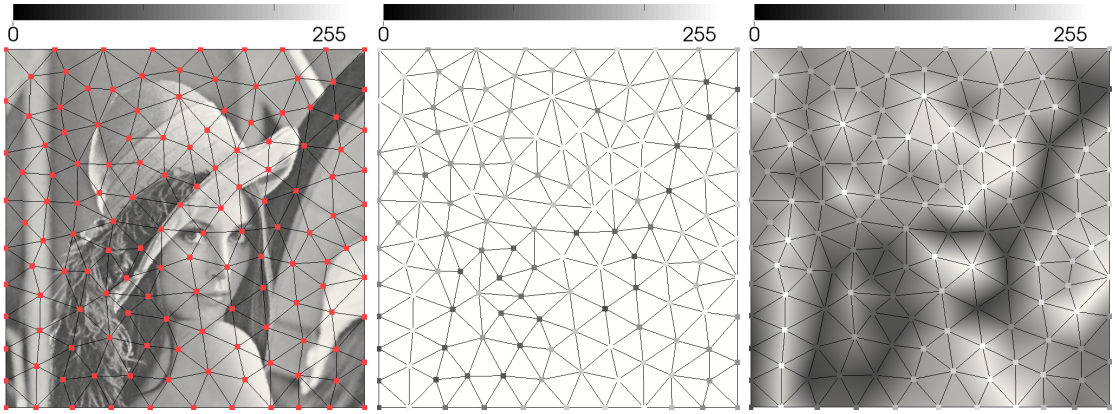
Figure 2.1: Illustration of the interpolation of a 2D image on a single triangular element: (a) image mapping; (b) nodal values, $u(\mathbf{X}^i)$; (c) Lagrange interpolation on the element, u_h.

Using this approach and enough elements, the mesh may approximately represent the original image. To outline our purpose, let us consider one other example, the 2D 8-bits image Lena [4], which is presented in Figure 2.2(a). This image is very popular in the image processing domain, containing $\widehat{T} = 262144$ pixels ($\widehat{L} \times \widehat{H} = 512 \times 512$). Firstly, we immerse this image into an uniform mesh, shown in Figure 2.2(b)), containing $N = 140$ nodes on a square of dimensions ($[0, 1] \times [0, 1]$). The result provides a rather coarse mesh size.

(a) Lena image, $\widehat{T} = 262144$ (b) Initial mesh, $N = 140$

Figure 2.2: "Lena" image [4] and initial mesh.

Like previously, the mesh nodes find the corresponding pixels, mapping the image, as seen in Figure 2.3(a). Therefore, the image information is stored in the nodes, Figure 2.3(b). The Lagrange interpolation solution of elements u_h is computed with the interpolation operator, from the nodes' value, also shown in Figure 2.3(c).

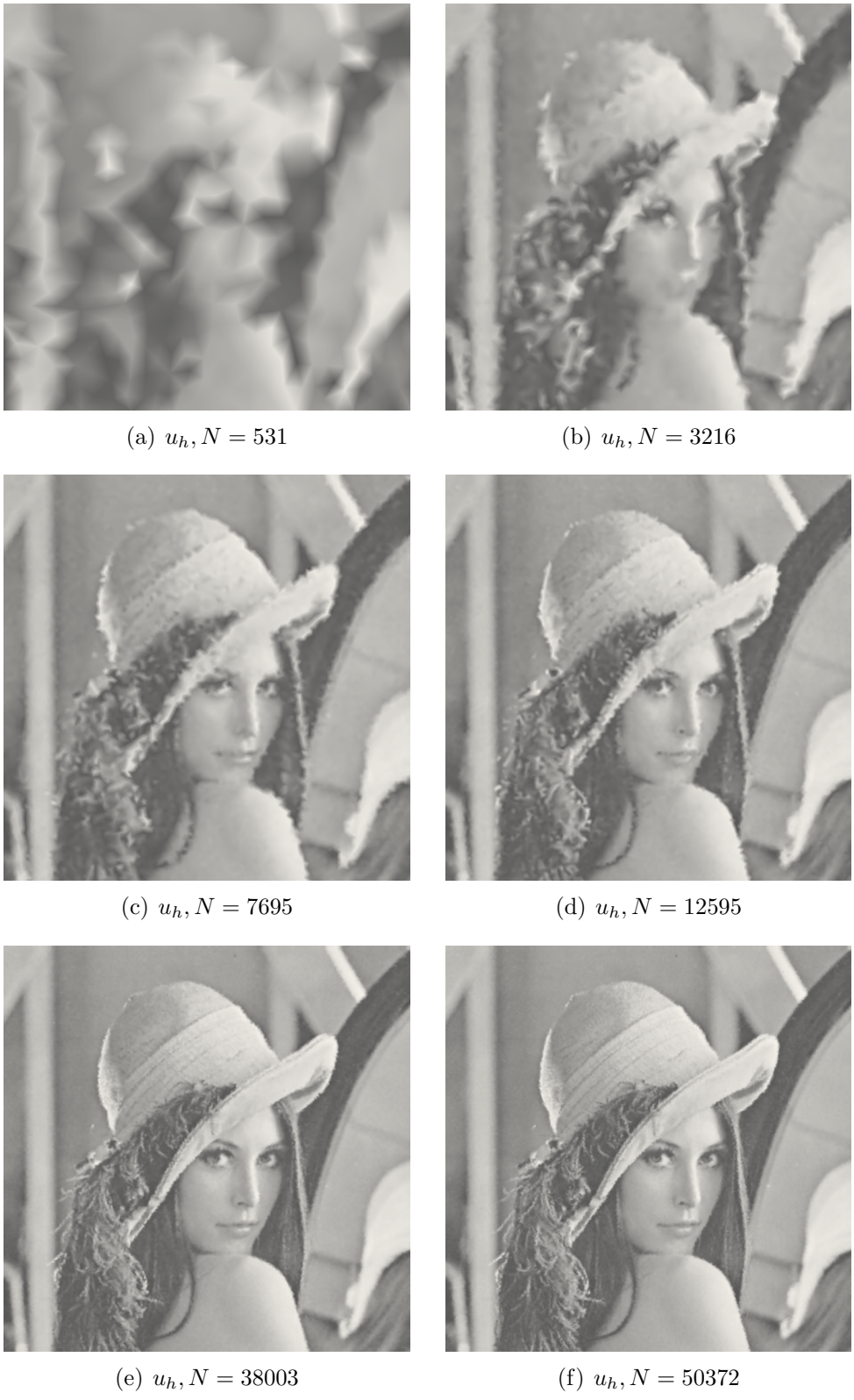


(a) Image map (b) $u(\mathbf{X}^i), N = 140$ (c) $u_h, N = 140$

Figure 2.3: Illustration of the interpolation of a 2D grey-scale image, "Lena" [4], on a mesh with 140 nodes: (a) node location on the image Lena; (b) nodal value, $u(\mathbf{x}^i)$; (c) Lagrange interpolation on the elements, u_h

We observe that, even if we recognize the global pattern, coarseness of the mesh results in a poor representation of the original image. Therefore, the simplest way to improve the representation is to increase the number of nodes or to reduce the size of the elements. Figure 2.4 shows the image interpolated on the mesh for an increasing number of nodes. The last one is very close to the original image, for a number of nodes still smaller than the number of pixels ($N = 50372$, $\widehat{T} = 262144$), reducing also the size of the storage of the information.

The Image Immersion method has also be extended to 3D image applications. To illustrate it, we have used a MRI of the human head from the BrainWeb Database [39], treated in [40, 41]. The 3D image corresponds to a healthy subject scanned with a T1-weighted contrast on a 1.5T magnetic field, with a 30^o flip angle, $22ms$ of repeat time, $9.2ms$ of echo time, and a $1mm$ isotropic voxel size, resulting in a $256 \times 256 \times 181$-sized volume. Additionally, ground-truth regions were also provided for the skin, skull, cerebrospinal fluid, grey and white matter tissues. To have a clearer visualization of the results, we have interpolated a sub-volume of $(91 \times 212 \times 181)$, with $\widehat{T} = 3491852$ voxels, as shown in Figure 2.5(a). The (sub)image contains half the human head regions referred before. The interpolation of the image on a 3D uniform mesh is drawn in Figures 2.5(b),(c) and (d) for different and increasing number of nodes.

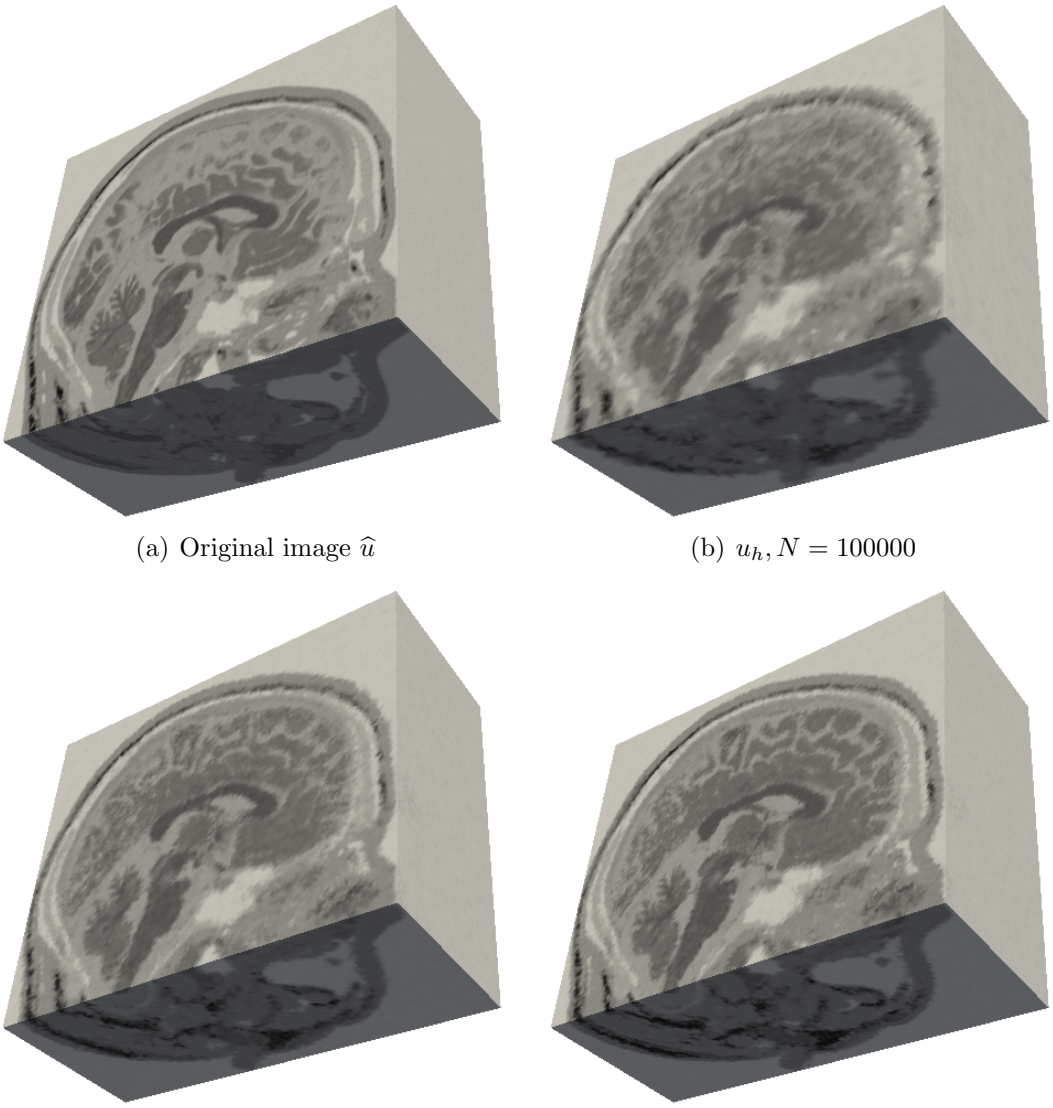


(a) $u_h, N = 531$ (b) $u_h, N = 3216$

(c) $u_h, N = 7695$ (d) $u_h, N = 12595$

(e) $u_h, N = 38003$ (f) $u_h, N = 50372$

Figure 2.4: Interpolation of the image "Lena" on a 2D mesh, using uniform meshes but with an increasing number of nodes.



(a) Original image $\widehat{u}$

(b) $u_h, N = 100000$

(c) $u_h, N = 600000$

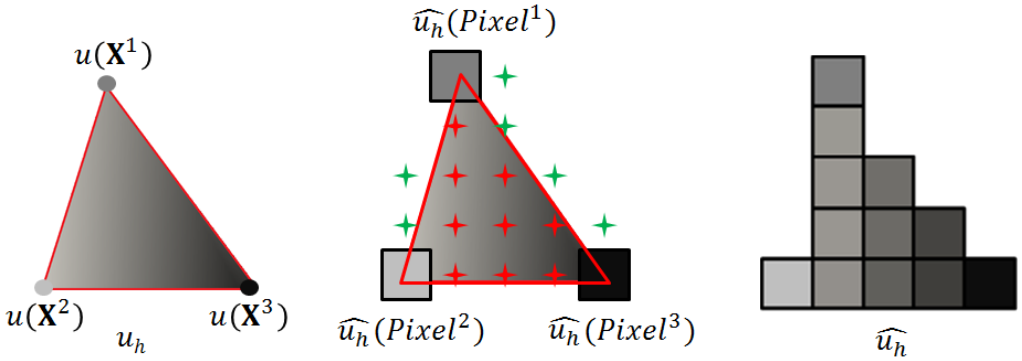
(d) $u_h, N = 1900000$

Figure 2.5: Application of the interpolation of the image on a 3D mesh, using uniform meshes and an increasing number of nodes.

2.3 Image construction and compression based on the mesh

The previous examples show that the Lagrange interpolation solution on the mesh may approximately represent the original image, often with fewer number of nodes than the number of pixels/voxels. It may help reducing the stored information, as in lossy image compression methods. To demonstrate this point and quantify the results, let us construct a new image, $\widehat{u_h}$, based on the resulting interpolation on the mesh, u_h, with the same number of pixels/voxels and dimension of the original image, $\widehat{T}$. This image construction procedure can be considered as the opposite operation of the previous methodology. Figure 2.6(a) shows again the interpolation solution on element K, with nodes $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3$. Firstly, the three nodes of element can directly create the three pixels, as given by Equation (2.8) and illustrated in Figure 2.6(b).

$$\begin{aligned} &\widehat{u_h}(Pixel^k/Voxel^k) = u_h(\mathbf{X}^i), \text{ where } \widehat{l^k} = \text{int}(\tfrac{x^i}{\mathrm{X}} \cdot (\widehat{L}-1)+1), \\ &\widehat{h^k} = \text{int}(\tfrac{y^i}{\mathrm{Y}} \cdot (\widehat{H}-1)+1), \underbrace{\widehat{w^k} = \text{int}(\frac{z^i}{\mathrm{Z}} \cdot (\widehat{W}-1)+1)}_{\text{only for 3D}} \ \forall\, i,\ldots,N \end{aligned} \tag{2.8}$$



(a) (b) (c)

Figure 2.6: Illustration of the image construction from the interpolation solution on the mesh: (a) interpolation on the element, u_h; (b) identification of the pixels' centers; (c) resulting image.

For the remaining pixels, their centers are identified in the element. If they are inside or along an edge, they are given by the values of the interpolation solution, u_h, in this center. The new created pixels construct the new image and are represented in Figure 2.6(c).

In the new image, $\widehat{u_h}$, with $\widehat{T}$ pixels/voxels, there are N pixels/voxels directly created from the nodes of the mesh, which are the same as the original image. For $N < \widehat{T}$, the created pixels/voxels may be different from the original ones. To research the quality of the results and the lossy image compression, we introduce the indicators MSE (Mean Square Error) and D (Density) as follows:

$$\begin{cases} MSE = \dfrac{1}{\widehat{T}} \displaystyle\sum_{k=1}^{\widehat{T}} (\widehat{u_h^k} - \widehat{u^k})^2 \\ \\ D = \dfrac{\text{Mesh size}}{\text{Original image size}} = \dfrac{\text{Number of nodes}}{\text{Number of pixels/voxels}} = \dfrac{N}{\widehat{T}} \end{cases} \tag{2.9}$$

They will allow, in the following, a comparison between the original and the created image, pixel by pixel or voxel by voxel.

2.4 Automatic anisotropic mesh construction

The other advanced way to improve the image representation, in particular for a constrained number of nodes, is to use an anisotropic mesh instead of an uniform one. Compared to uniform meshes, elements of anisotropic ones have different shapes, sizes and orientations, which may improve the accuracy of the numerical simulation solution, especially in the case where there are discontinuities or high gradients of the solution. It enables to capture physical phenomena such as boundary layers, shock waves or moving interfaces [42, 29, 43, 44, 37]. In our application, anisotropic meshes may also improve the interpolation solution with fewer number of nodes.

In this section, we have improved and applied the methodology presented in [28, 37, 38], which is based on a topological mesh generator that has as input a nodal metric map, being the metric tensor constructed from an edge based *a posteriori* error estimator. This error estimate is computed by building the length distribution tensor and the detail of this error analysis has been presented in [37]. For this remeshing procedure, a stretching factor is applied on each edge to obtain the corresponding metric tensor, with a constrained number of nodes. Furthermore, the interpolation error computation can be applied and extended to other components or multi-component situations such as color images, velocity or pressure fields, ...

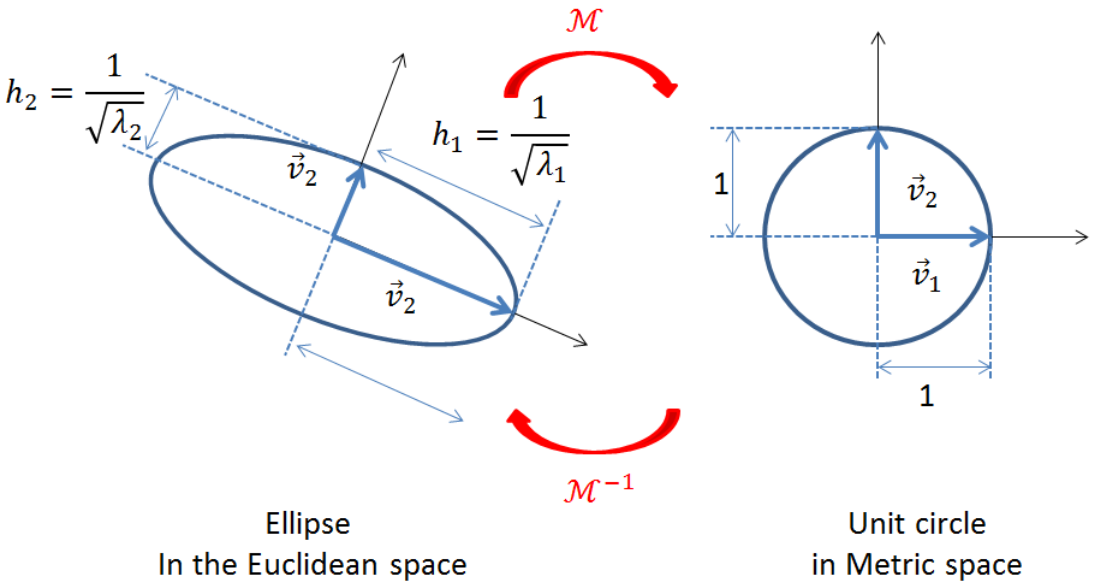
2.4.1 Mesh adaptation based on a metrics field

Anisotropic mesh generation techniques based on metric tensors have been developed in the last decade [28, 45, 44]. The metric-based method is not established on the Euclidean space, but on the Riemannian metric space and on the unit mesh (for example, equilateral triangles in a 2D mesh or regular tetrahedron in 3D meshes). In other words, the mesh can be composed of elements of any shape (unit or not) in the Euclidean space, which will be transformed into unit elements in the given metric tensor space.

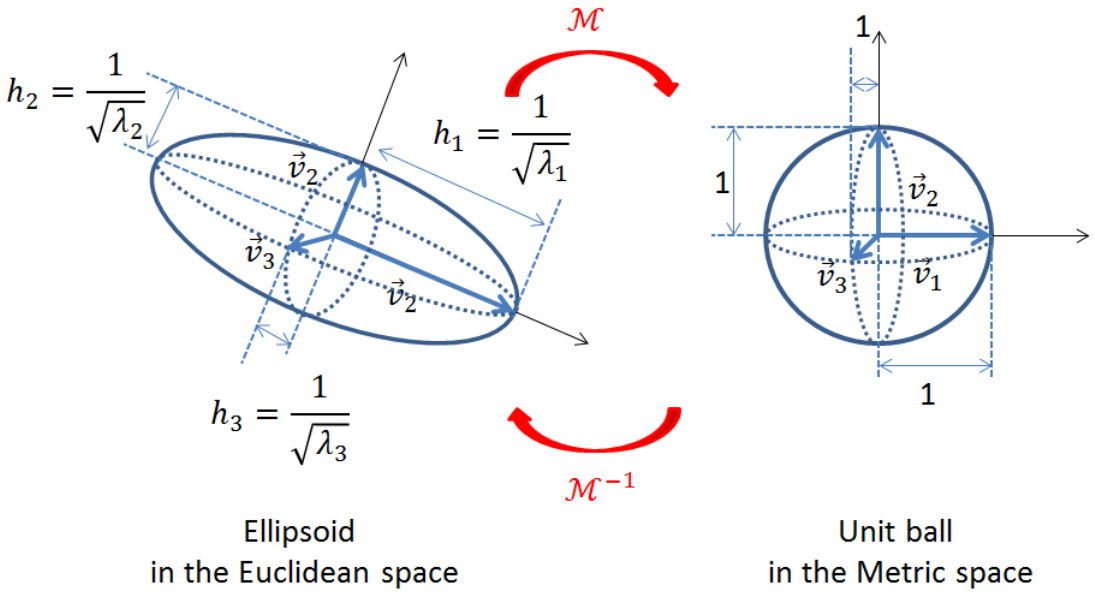
We note the Riemannian metric tensor as $\mathcal{M}$ in $\mathbb{R}^n$ and is a $n \times n$ symmetric positive matrix. It is diagonalizable and its associated eigenvectors are $\mathcal{R}$ (as a rotation vector), being Λ the metric of its eigenvalues, as follows:

$$\mathcal{M} = {}^t\mathcal{R}\Lambda\mathcal{R} = {}^t\mathcal{R}\begin{bmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{bmatrix}\mathcal{R}, \text{ in a 3D mesh} \tag{2.10}$$

The metric tensor may be represented by an ellipse (in 2D) or an ellipsoid in 3D, as seen in Figure 2.7, where $h_1, h_2, \ldots, h_n$ are the local sizes in each direction of the metric tensor $\mathcal{M}$.



(a)



(b)

Figure 2.7: Representation of the ellipse (2D) and ellipsoid (3D) transformation into unit circle and unit ball from the Euclidean space to the metric space.

The scalar product of two vectors in the Euclidean space and in the Riemannian

metric space are written:

$$(\vec{\mathbf{u}}, \vec{\mathbf{v}}) = {}^t\mathbf{u}\mathbf{v} \text{ and } (\vec{\mathbf{u}}, \vec{\mathbf{v}})_{\mathcal{M}} = {}^t\mathbf{u}\mathcal{M}\mathbf{v} \in \mathbb{R}^n \tag{2.11}$$

For this reason, the norm of the vector $\vec{\mathbf{u}}$ in the Riemannian metric space is:

$$\|\vec{\mathbf{u}}\|_{\mathcal{M}} = \sqrt{(\vec{\mathbf{u}}, \vec{\mathbf{u}})_{\mathcal{M}}} = \sqrt{{}^t\mathbf{u}\mathcal{M}\mathbf{u}} \tag{2.12}$$

It means that, for each edge of the mesh, its length in the Riemannian metric space is always equal to the unit length. The definitions of the Euclidean distance and of the Riemannian distance, corresponding to vector $\mathbf{X}^{ij}$ between nodes $\mathbf{X}^i$ and $\mathbf{X}^j$, are given as:

$$\mathrm{d}(\mathbf{X}^{ij}, \mathbf{X}^{ij}) = \sqrt{{}^t\mathbf{X}^{ij}\mathbf{X}^{ij}} \text{ and } \mathrm{d}(\mathbf{X}^{ij}, \mathbf{X}^{ij})_{\mathcal{M}} = \sqrt{{}^t\mathbf{X}^{ij}\mathcal{M}\mathbf{X}^{ij}} = 1 \tag{2.13}$$

The metric is continuous along the edge. The tensor may be defined along the vector $\mathbf{X}^{ij}$, as a function of the position of a point x, such that $\mathbf{X}^{ij}(x) = \mathbf{X}^i + t\mathbf{X}^{ij}, t \in [0, 1]$. Thus,

$$\mathrm{d}(\mathbf{X}^{ij}(x), \mathbf{X}^{ij}(x))_{\mathcal{M}} = \int_0^1 \sqrt{{}^t\mathbf{X}^{ij}\mathcal{M}\left(\mathbf{X}^i + t\mathbf{X}^{ij}\right)\mathbf{X}^{ij}}dt \tag{2.14}$$

For the anisotropic mesh construction, the metric field may be associated with the mesh elements. For example, a triangle or a tetrahedron becomes an equilateral triangle or regular tetrahedron in the metric space, as seen in as Figure 2.8.

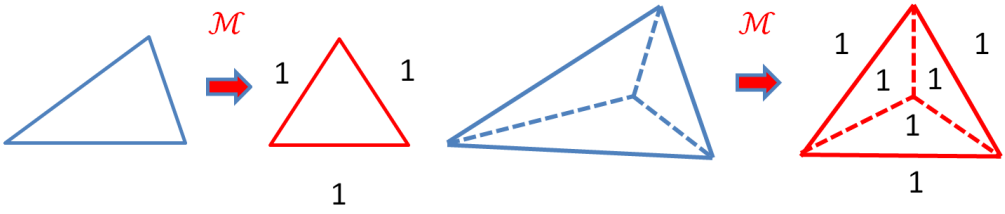
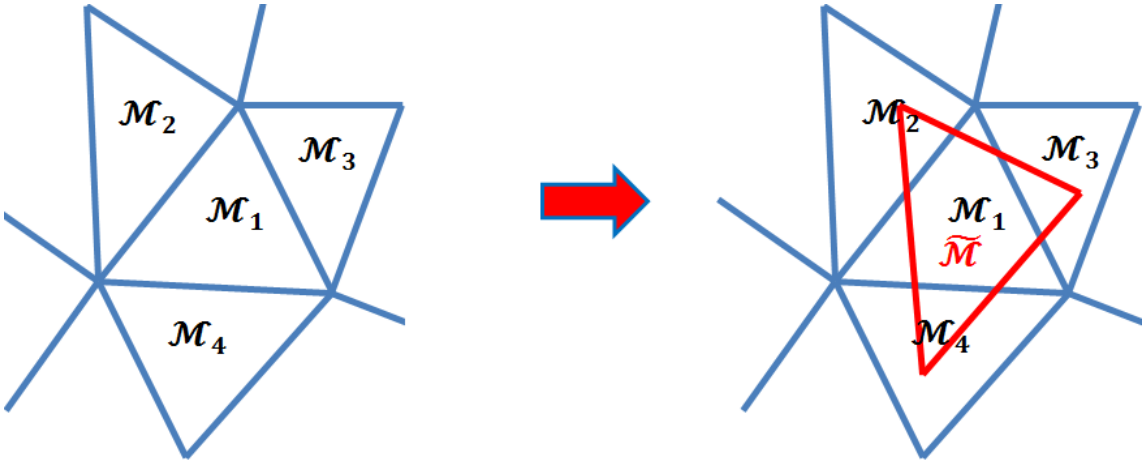


Figure 2.8: Unit metric for a 2D and a 3D elements.

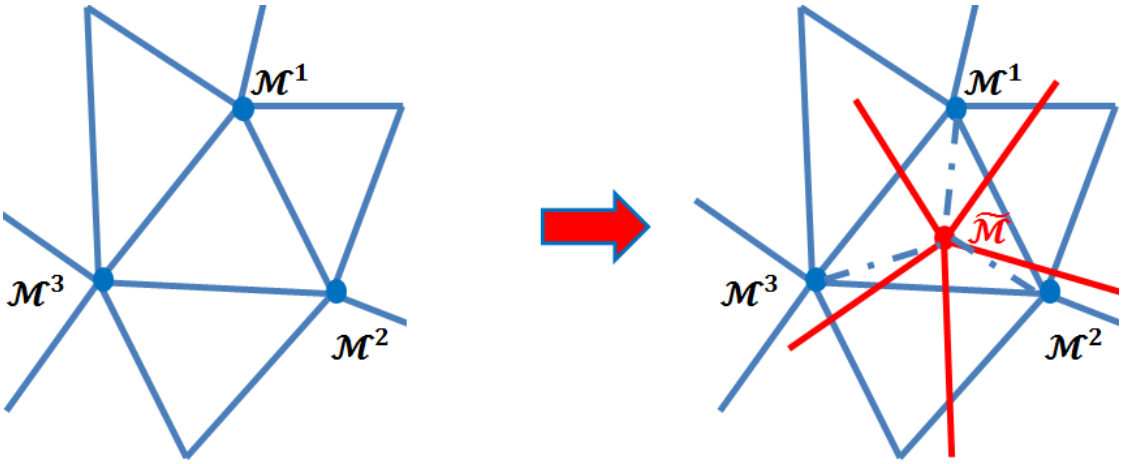
In this way, the metric tensor is stored at the element, and is piecewise constant along the edge (Equation (2.14)), but discontinuous from element to element. For the iterative mesh adaptation procedure, using the metric tensor at the element, as seen in Figure 2.9(a), is not the best choice to obtain the new metric tensor $\widetilde{\mathcal{M}}$ at the new element, since it involves four parts of four different metric tensors, $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$.

Indeed, the nodal metric tensors computation has been most widely used [29, 44, 37]. The metric tensor in the new point $\widetilde{\mathbf{X}}$, which is the isobarycenter of $\{\mathbf{X}^1, \ldots, \mathbf{X}^n\}$, is given by:

$$\widetilde{\mathcal{M}} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{M}^i \tag{2.15}$$



(a) Metric on elements



(b) Metric on nodes

Figure 2.9: Illustration of mesh adaptation with a metric field defined at the elements and at the nodes.

where n is the number of nodes around the new node. One way to build this nodal value is also to perform a metric intersection method proposed and applied in the literature [46, 42, 47, 44], for the case where we have several meshes. Let us consider the two metric $\mathcal{M}_1$-$\mathcal{M}_2$ intersection and the matrix $\mathcal{M}_{12} = \mathcal{M}_1^{-1}\mathcal{M}_2$ with the normalized eigenvectors e_1, e_2, e_3. This latter are a basis of the Riemannian space $\mathbb{R}^3$. If we define $\mathcal{P} = (e_1\ e_2\ e_3)$, then $\mathcal{M}_1$ and $\mathcal{M}_2$ are

$$\mathcal{M}_1 = {}^t\mathcal{P}^{-1} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \mathcal{P}^{-1} \text{ and } \mathcal{M}_2 = {}^t\mathcal{P}^{-1} \begin{pmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{pmatrix} \mathcal{P}^{-1} \tag{2.16}$$

The final intersected metric $\mathcal{M}_{1\cap 2} = \mathcal{M}_1 \cap \mathcal{M}_2$ is:

$$\mathcal{M}_{1\cap 2} = {}^t\mathcal{P}^{-1} \begin{pmatrix} \max(\lambda_1, \mu_1) & 0 & 0 \\ 0 & \max(\lambda_2, \mu_2) & 0 \\ 0 & 0 & \max(\lambda_3, \mu_3) \end{pmatrix} \mathcal{P}^{-1} \tag{2.17}$$

For more than two metric tensors intersection, authors consider generally metrics that are intersected two by two.

$$\mathcal{M}_{\cap_i \mathcal{M}_i} = (((\mathcal{M}_1 \cap \mathcal{M}_2) \cap \mathcal{M}_3) \ldots) \cap \mathcal{M}_k \tag{2.18}$$

This intersection method requires firstly all metric tensors construction to then obtain the final intersected metric tensor, which may require high computation resources. One other method to compute only one metric tensor at the node is proposed by [37], by skipping intersection computation and will be applied in the following anisotropic mesh adaptation procedure. Firstly, let us recall that $\mathbf{X}^i \in \mathbb{R}^d$, $i = 1, \cdots, N$ is the set of nodes of the mesh and that the vector $\mathbf{X}^{ij}$ connects nodes i and j, $\mathbf{X}^{ij} = \mathbf{X}^j - \mathbf{X}^i$, as illustrated in Figure 2.10.

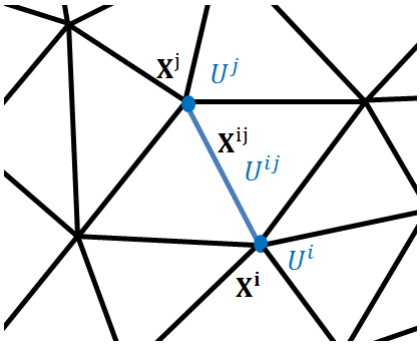


Figure 2.10: Illustration of the edge vector $\mathbf{X}^{ij}$ of the edge joining nodes i and j, and the edge solution, U^{ij}, joining the solution at the nodes i and j.

In the mesh, one node can be shared by several edges. The set of nodes connected to node i is $\Gamma(i)$, given by:

$$\Gamma(i) = \left\{ j \,,\, \exists K \in \mathcal{K} \,,\, \mathbf{X}^i, \mathbf{X}^j \text{ are nodes of } K \right\} \tag{2.19}$$

Let each edge vector connected to node $\mathbf{X}^{ij}$ be transformed in an unit length vector in the Riemannian space. One may have different element metric tensors that satisfy the required conditions, since the edges are shared by the latter. In fact,

$$\mathrm{d}(\mathbf{X}^{ij}, \mathbf{X}^{ij})_{\mathcal{M}^{ij}} = \left(\mathcal{M}^{ij}\mathbf{X}^{ij}, \mathbf{X}^{ij}\right) = 1 \text{ for } j \in \Gamma(i) \tag{2.20}$$

However, we intend to build one new metric tensor, $\mathcal{M}^i$, at node i instead of these several metric tensors, where each transformed edge vector is almost equal to 1, so that the sum of the transformed edge vectors in $\Gamma(i)$ is:

$$\sum_{j \in \Gamma(i)} \left(\mathcal{M}^i \mathbf{X}^{ij}, \mathbf{X}^{ij}\right) = \sum_{j \in \Gamma(i)} 1 = |\Gamma(i)| \tag{2.21}$$

In the following, we use the tensor scalar product notation:

$$\mathcal{A} \otimes \mathcal{B} = \begin{pmatrix} A_{11}B_{11} & A_{12}B_{12} \\ A_{21}B_{21} & A_{22}B_{22} \end{pmatrix} \tag{2.22}$$

Equation (2.21) can be written as:

$$\mathcal{M}^i : \left(\sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right) = |\Gamma(i)| \tag{2.23}$$

In fact, it is very difficult to find a metric tensor, which guarantees that each edge may be transformed to one with unit length. However, we may accept a compromised way by computing

$$\mathcal{M}^i = \underset{\mathcal{M}^i \in \mathbb{R}^{d \times d}}{\operatorname{argmin}} \left(\sum_{j \in \Gamma(i)} \left(\left\| \mathbf{X}^{ij} \right\|^2_{\mathcal{M}^i} - 1 \right)^2 \right) \tag{2.24}$$

In [37], the author proposed a new definition called the length distribution tensor at node i, written as

$$\mathcal{X}^i = \frac{1}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \tag{2.25}$$

It will be used to find the optimized metric tensor because the latter, evaluated at node i, may be constructed as a function of $\mathcal{X}^i$, as follows:

$$\mathcal{M}^i = \frac{1}{d} (\mathcal{X}^i)^{-1} \tag{2.26}$$

where d is the spatial dimension. In addition, the length distribution tensor will be further used, for example in gradient recovery procedures, as we will see later.

2.4.2 Edge based error estimation

The objective of mesh adaptation is to construct a mesh which is optimal for the simulations to be performed. In other words, it should ensure that the estimated error between the exact solution and the approximate one is minimal.

First, let us recall the classical method to compute the interpolation error estimate. u denotes the regular scalar field, U^i being its value at node $\mathbf{X}^i$, and u_h is the interpolation result on the given mesh, with N nodes and using the Lagrange interpolation operator from $\mathcal{V}$ to $\mathcal{V}_h$.

Even if the gradient of u_h is a piecewise constant vector field, $\nabla u \in \mathcal{C}^1(\Omega)$. The projection of u_h along the edges is continuous and one may write:

$$U^j = U^i + \nabla u_h \cdot \mathbf{X}^{ij} \text{ and } \nabla u_h \cdot \mathbf{X}^{ij} = U^{ij} \tag{2.27}$$

Using the analysis carried out in [37], we can set that the norm of the difference between the projection of the gradient along the edges and the interpolated value of the gradient can be upper bounded by:

$$\left\| \nabla u_h \cdot \mathbf{X}^{ij} - \nabla u(\mathbf{X}^i) \cdot \mathbf{X}^{ij} \right\| \leq \max_{y \in \left[\mathbf{X}^i, \mathbf{X}^j \right]} \left| \mathbb{H}(u)(y) \mathbf{X}^{ij} \cdot \mathbf{X}^{ij} \right| \tag{2.28}$$

where $\mathbb{H}(u) = \nabla^2 u$ is the associated Hessian of u. For that, the second derivative of u is necessary and its projected value can be established using Equation (2.27) and the interpolation operator, to write a second order derivation as:

$$\left[\nabla(\nabla u_h)\mathbf{X}^{ij}\right]\cdot\mathbf{X}^{ij}=\nabla u(\mathbf{X}^{ij})\cdot\mathbf{X}^{ij} \tag{2.29}$$

The Taylor series of the gradient of u is:

$$\nabla u(\mathbf{X}^j)=\nabla u(\mathbf{X}^i)+\mathbb{H}(u)(\mathbf{X}^i)\mathbf{X}^{ij} \tag{2.30}$$

Then, projecting onto $\mathbf{X}^{ij}$ gives:

$$\left(\nabla u(\mathbf{X}^j)-\nabla u(\mathbf{X}^i)\right)\cdot\mathbf{X}^{ij}=\mathbb{H}(u)(\mathbf{X}^i)\mathbf{X}^{ij}\cdot\mathbf{X}^{ij} \tag{2.31}$$

We use this result to approximate the error along the edge as

$$e^{ij}=\left|\left[\nabla u(\mathbf{X}^j)-\nabla u(\mathbf{X}^i)\right]\cdot\mathbf{X}^{ij}\right| \tag{2.32}$$

However, a node is always shared by different edges and elements, and the gradient value of u at the node may be different when computed from different edges separately. Inspired from the length distribution tensor presented in the previous section, we find a recovered gradient value of u at node $\mathbf{X}^i$, noted $\mathbf{G}^i$, proposed by [37] and satisfying the following condition:

$$\mathbf{G}^i=\underset{\mathbf{G}}{\operatorname{argmin}}\left(\sum_{j\in\Gamma(i)}\left|\left(\mathbf{G}-\nabla u(\mathbf{X}^{ij})\right)\cdot\mathbf{X}^{ij}\right|^2\right)=\underset{\mathbf{G}}{\operatorname{argmin}}\left(\sum_{j\in\Gamma(i)}\left|\mathbf{G}\cdot\mathbf{X}^{ij}-\mathbf{U}^{ij}\right|^2\right) \tag{2.33}$$

so that,

$$0=\sum_{j\in\Gamma(i)}\left(\mathbf{G}^i\left(\mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right)-\mathbf{U}^{ij}\mathbf{X}^{ij}\right) \tag{2.34}$$

According to the previous definition of the length distribution tensor $\mathcal{X}^i$, the recovered gradient $\mathbf{G}^i$ at node $\mathbf{X}^i$ is given by:

$$\mathbf{G}^i=\left(\mathcal{X}^i\right)^{-1}\mathcal{U}^i \text{ and } \mathcal{U}^i=\frac{1}{\Gamma(i)}\sum_{j\in\Gamma(i)}\mathbf{U}^{ij}\mathbf{X}^{ij} \tag{2.35}$$

We set $\mathbf{G}^{ij}=\mathbf{G}^j-\mathbf{G}^i$ as the gradient vector along the edge $\mathbf{X}^{ij}$, which follows

$$\mathbf{G}^{ij}\cdot\mathbf{X}^{ij}=\mathbb{H}(u)(\mathbf{X}^i)\mathbf{X}^{ij}\cdot\mathbf{X}^{ij} \tag{2.36}$$

According to Equation (2.32), the new estimated error is then a function of the recovered gradient $\mathbf{G}$ as

$$e^{ij}=\left|\mathbf{G}^{ij}\cdot\mathbf{X}^{ij}\right|=\left|\mathbb{H}(u)(\mathbf{X}^i)\mathbf{X}^{ij}\cdot\mathbf{X}^{ij}\right| \tag{2.37}$$

The error tensor $\mathcal{E}^i$ at node $\mathbf{X}^i$ can be written as:

$$\mathcal{E}^i=\underset{i}{\operatorname{argmin}}\sum_{j\in\Gamma(i)}\left|\mathcal{E}^i\cdot\mathbf{X}^{ij}-e^{ij}\mathbf{X}^{ij}\right|^2 \tag{2.38}$$

Finally, the nodal error E^i is a function of the length distribution tensor $\mathcal{X}^i$ as follows:

$$E^i = \det(\mathcal{E}^i) = \det\left(\left(\mathcal{X}^i\right)^{-1} \sum_{j\in\Gamma(i)} e^{ij}\mathbf{X}^{ij} \otimes \mathbf{X}^{ij}\right) \tag{2.39}$$

2.4.3 Metric construction with control of the number of nodes

In this section, we explain how to link this estimated error to the new anisotropic metric field construction. In order to re-generate the mesh properly based on an existing one, it is necessary to compute the new length for each edge, by applying a stretching factor s^{ij} to the edge, so that

$$\widetilde{\mathbf{X}^{ij}} = s^{ij}\mathbf{X}^{ij} \tag{2.40}$$

The stretch factor of each edge, s^{ij}, will then depend on the estimated error e^{ij} and the target error $\overline{e^{ij}}$, wanted on the edge^{ij}, such that:

$$(s^{ij})^2 = \frac{\overline{e^{ij}}}{e^{ij}} \tag{2.41}$$

If the estimation error is larger than the target error then the edge length needs to be shrinked, $s^{ij} < 1$. At the opposite, if the estimated error is smaller than the target one, the edge length is stretched and $s^{ij} > 1$.

However, during the mesh adaptation procedure, controlling the number of nodes of the mesh is required to avoid reaching too fine meshes and to be optimal in terms of the computational cost. The stretch factor s^{ij} used to scale the edge changes quadratically, Equation (2.41). Hence, let n^{ij} be the number of created nodes. Its relation with the stretching factor s^{ij} is given by:

$$n^{ij} = (s^{ij})^{-1} = \left(\frac{\overline{e^{ij}}}{e^{ij}}\right)^{-\frac{1}{2}} \tag{2.42}$$

This allows us to control the number of created nodes along the different edge directions at node i. By processing in a similar way as for the length distribution tensor, [38] gives a distribution of nodes tensor, $\mathcal{N}^i$, at the node $\mathbf{X}^i$, which is the solution of an optimization problem:

$$\mathcal{N}^i = \underset{i}{\operatorname{argmin}} \sum_{j\in\Gamma(i)} \left|\mathcal{N}^i \cdot \mathbf{X}^{ij} - n^{ij}\mathbf{X}^{ij}\right|^2 \tag{2.43}$$

so that,

$$0 = \mathcal{N}^i : \sum_{j\in\Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} - \sum_{j\in\Gamma(i)} n^{ij}\mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \tag{2.44}$$

The total number of created nodes per node i is then $\mathbf{N}^i = \det(\mathcal{N}^i)$,

$$\mathbf{N}^i = \det(\mathcal{N}^i) = \det\left(\left(\mathcal{X}^i\right)^{-1} \sum_{j\in\Gamma(i)} n^{ij}\mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right) \tag{2.45}$$

By considering an averaging process of the number of nodes distribution function, the total given number of nodes N in the adapted mesh will be

$$N = \sum_i \mathbf{N}^i \tag{2.46}$$

Assuming that we will impose an uniform totally balanced error along the edges $\overline{e^{ij}} = e =$ constant, the number of created nodes per edge is:

$$n^{ij}(e) = (s^{ij}(e))^{-1} = (\frac{e}{e^{ij}})^{-\frac{1}{2}} \tag{2.47}$$

The equation of the total number of created nodes at node i is:

$$\mathbf{N}^i(e) = \det\left(\left(\mathcal{X}^i\right)^{-1} \sum_{j\in\Gamma(i)} (\frac{e}{e^{ij}})^{-\frac{1}{2}}\mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right) \tag{2.48}$$

Treatment of this equation gives:

$$\mathbf{N}^i(e) = e^{-\frac{d}{2}}\det\left(\left(\mathcal{X}^i\right)^{-1} \sum_{j\in\Gamma(i)} (\frac{1}{e^{ij}})^{-\frac{1}{2}}\mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right) \tag{2.49}$$

Let us consider $\mathbf{N}^i(1)$ as:

$$\mathbf{N}^i(1) = \det\left(\left(\mathcal{X}^i\right)^{-1} \sum_{j\in\Gamma(i)} (\frac{1}{e^{ij}})^{-\frac{1}{2}}\mathbf{X}^{ij}\otimes\mathbf{X}^{ij}\right) \tag{2.50}$$

then,

$$\mathbf{N}^i(e) = e^{-\frac{d}{2}}\mathbf{N}^i(1) \tag{2.51}$$

The total number of nodes in the new mesh is

$$N = e^{-\frac{d}{2}}\sum_i \mathbf{N}^i(1). \tag{2.52}$$

Therefore, the global induced error for a given fixed number of nodes N can be written as:

$$e(N) = \left(\frac{N}{\sum\limits_i \mathbf{N}^i}\right)^{-\frac{2}{d}} \tag{2.53}$$

The target error of edge $\overline{e^{ij}}$ can be replaced by $e(N)$ using this approach. The stretching factors to be computed in order to obtain the new metric field, under the

constraint of a fixed number of nodes N and an uniform balanced error along the edges, e, is thus given by

$$s^{ij} = \left(\frac{e(N)}{e^{ij}} \right)^{\frac{1}{2}} \tag{2.54}$$

The new metric tensor only depends now of the given fixed number of nodes N, the estimation error e^{ij} and the length distribution tensor $\mathcal{X}^i$.

One may simply define the new length distribution tensor as

$$\widetilde{\mathcal{X}^i} = \frac{1}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} \widetilde{\mathbf{X}^{ij}} \otimes \widetilde{\mathbf{X}^{ij}} = \frac{1}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} (s^{ij})^2 \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \tag{2.55}$$

as well as the new metric to give to the mesher:

$$\widetilde{\mathcal{M}^i} = \frac{1}{d} (\widetilde{\mathcal{X}^i})^{-1} \tag{2.56}$$

2.4.4 Extension to multiphase field adaptation

A new way to construct a unique metric from a multi-component vector field has been introduced in [38] and used instead of classical intersection techniques. This vector can be composed of the velocity, pressure, multi-level-set functions or other fields composing $\vec{u} = \{u_1, u_2, \ldots, u_n\}$. We may then define the error vector for all these components, $\overrightarrow{e^{ij}} = \left\{ e_1^{ij}, e_2^{ij}, \ldots, e_n^{ij} \right\}$. Then, the stretch factor is

$$s^{ij} = \left(\frac{\left\| \widetilde{e^{ij}} \right\|}{\left\| \overrightarrow{e^{ij}} \right\|} \right)^{\frac{1}{2}} \tag{2.57}$$

where the error norm can be L_2 or L_∞, to give

$$\underbrace{\left\| e^{ij} \right\| = \left(\sum_{k=1}^{n} \left(e_k^{ij} \right)^2 \right)^{\frac{1}{2}}}_{L_2} \quad \text{or} \quad \underbrace{\left\| e^{ij} \right\| = \max_{k=1,\ldots,n} e_k^{ij}}_{L_\infty} \tag{2.58}$$

To illustrate this point, let us suppose v_h the finite element solution of a multi-phase Navier-Stokes problem, which needs the velocity vector field $\overrightarrow{V}$ and a phase function α. Let $v_h(\mathbf{X}^i) = V^i$ and we introduce the vector field $\mathcal{Y} = \left(\frac{V}{|V|}, |V|, \alpha \right)$. We obtain, for each node i,

$$\Pi_h \mathcal{Y}(\mathbf{X}^i) = \left\{ \frac{V^i}{|V^i|}, \left| V^i \right|, \alpha \right\} = \mathcal{Y}^i \tag{2.59}$$

The particular case of $|v| = 0$ is accounted using $\frac{V^i}{\max(|V^i|, \varepsilon)}$, with $\varepsilon = 10^{-6}$ being a small value, so that $\mathcal{Y}_k^i = 0$ when $|V^i| = 0$. In our case, we have focused mainly on the change of direction rather than on the intensity of the velocity.

2.4.5 MTC mesh generator

The topological mesh generator "**MTC**" [27, 48, 28, 29] has been presented in the previous chapter, showing that an initial mesh is thus iteratively modified until a local optimized mesh enforces two main criteria: the **minimal volume** and the **geometrical quality**.

In anisotropic adaptation, the second criterion, the **geometrical quality** q_K, is evaluated for each element using the metric field.

As an example, let us consider an initial mesh and a given metric field, $\mathcal{M} \in \mathcal{R}^{d\times d}$, which may be computed by the presented metric tensor construction method with an estimated error. Therefore, the quality $q(K)$ of the element measured in the metric $\mathcal{M}$ is defined as the minimum of two subcriteria: the **quality**, $C_q(K)$, and the **size**, $C_s(K)$.

$$q(K) = \min\left(C_q(K), C_s(K)\right) \tag{2.60}$$

The quality criterion, $C_q(K)$, is computed as:

$$C_q(K) = c_0 \frac{\text{Volume}(K)_{\mathcal{M}_K}}{h^d_{\mathcal{M}_K}} \tag{2.61}$$

where d is the spatial dimension and the matrix $\mathcal{M}_K$ of the metric of the element K is calculated as the average of the nodal metric matrix on the nodes of element K:

$$\mathcal{M}_K = \frac{1}{d+1}\sum_{i\in K}\mathcal{M}^i \tag{2.62}$$

$\text{Volume}(K)_{\mathcal{M}_K}$ is the volume of the transformed element K in the metric space $\mathcal{M}_K$, given by:

$$\text{Volume}(K)_{\mathcal{M}_K} = \text{Volume}(K)\sqrt{\det(\mathcal{M}_K)} = \int_{K_{\mathcal{M}_K}} dK_{\mathcal{M}_K} \tag{2.63}$$

and c_0 is the normalization factor, to ensure that Equation (2.61) gives a quality equal to 1 when the element K is equilateral in the metric tensor space, $\mathcal{M}_K$. Also,

$$c_0 = \frac{d!}{\sqrt{d+1}} 2^{d/2} \tag{2.64}$$

so that $c_0 = 4/\sqrt{3}$ for a 2D equilateral triangle, and $c_0 = 6\sqrt{2}$ for a 3D regular tetrahedron. $h_{\mathcal{M}_K}$ is the average of the lengths of the edges of element K transformed in the metric tensor $\mathcal{M}_K$:

$$h_{\mathcal{M}_K} = \left(\frac{2}{d(d+1)}\sum_{(i,j)\in K}(\mathcal{M}_K\mathbf{X}^{ij}, \mathbf{X}^{ij})\right)^{1/2} \tag{2.65}$$

The value of $C_q(K)$ varies from 0 to 1. Then, the second factor, the **size** criterion $C_s(K)$ controls the size of the element in the metric $\mathcal{M}_K$, as:

$$C_s(K) = \min\left(\frac{1}{h_{\mathcal{M}_K}}, h_{\mathcal{M}_K}\right)^d \tag{2.66}$$

which also varies between 0 and 1. Under the definition of the quality of the elements $q(K)$, we need to compare the different sets (like in Figure 1.12) in order to choose the best local cavity with the highest quality.

In conclusion, the minimal volume assures the conformity of the mesh, if the initial mesh was conforming, the latter handles improvements of element shape, size, connectivity, etc, depending on the quality function q_K, function of the geometry of the element K and of the prescribed background metric, which give together a measure for the element size and shape (aspect ratio) based on the computed metric tensor field.

2.4.6 Numerical tests

In this section, the image to mesh interpolation procedure is coupled to automatic anisotropic mesh adaptation with the constraint of a given number of nodes. At each iteration, the original image $\widehat{u}$ is interpolated in the current mesh, to build u. Then, an error estimator computes the estimated error along each edge, e^{ij}. On the other hand, a target error, $e(N)$, is defined as a function of an imposed number of nodes. Finally, computation of a stretch factor for each edge, s^{ij}, is performed. From this, a new metric tensor field $\widetilde{\mathcal{M}}$ is established, which is used to construct the new adapted anisotropic mesh. In this process, the mesh generator only needs the wanted number of nodes. The initial mesh does not influence the final results, and may be a very coarse mesh. The different steps of the whole Immersed Image Method are summarized in the algorithm presented below.

Input: Initial image $\widehat{u}$, initial mesh $\mathcal{H}$ and wanted number of nodes N
Output: Adapted anisotropic mesh $\widetilde{\mathcal{H}}$ with N nodes, solution interpolated u_h on this mesh.

1 **while** *the adapted anisotropic mesh is not achieved* **do**
2 Interpolate the image $\widehat{u}$ on the current mesh, to obtain u_h.
3 Compute the recovered gradient $\mathbf{G}$ on each node $\mathbf{G}^i$.
4 Estimate the error, e^{ij}, for the each edge $\mathbf{X}^{ij}$.
5 Compute the target error as a function of the given number of nodes, $e(N)$.
6 Build the stretch factor, s^{ij}, using the estimated error, e^{ij} and the target error, $e(N)$.
7 Construct the new metric tensor field, $\widetilde{\mathcal{M}}$.
8 Generate the optimal mesh, $\widetilde{\mathcal{H}}$, using $\widetilde{\mathcal{M}}$, and following the minimal volume and the geometrical quality criteria.
9 Update the mesh.

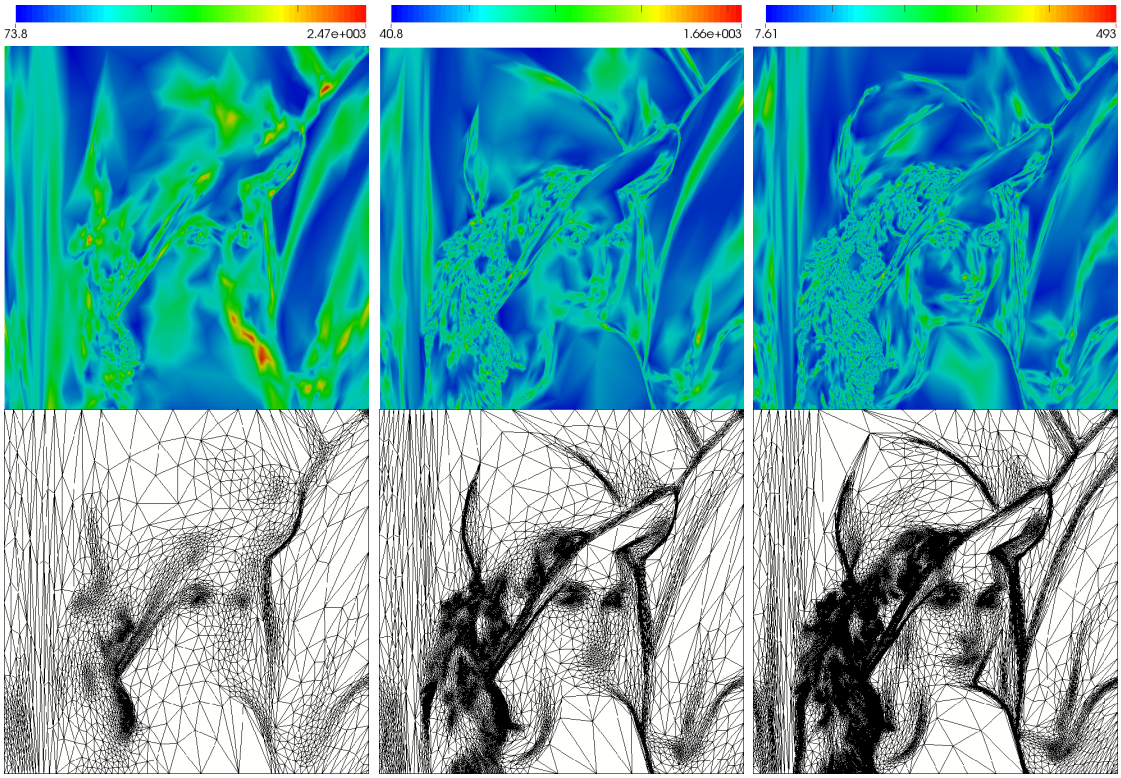
2.4.6.1 Application to "Lena" image

In a previous example, the "Lena" image (Figure 2.2(a)) has been used to illustrate the image interpolation procedure. Now, this procedure has been coupled to automatic anisotropic adaptation. Let us consider different number of wanted nodes, N: 3200, 13000, 35000. After 15 iterations on one core (3.5GHz,16 Gb RAM), the anisotropic mesh adaptation reaches convergence (the mesh no longer evolves). The CPU time (s) for each case is given in Table 2.1.

N	3200	13000	35000
Time (s)	35	99	251

Table 2.1: CPU time (s) for image interpolation coupled to mesh adaptation, for an increasing number of nodes and 15 iterations, runs performed on one core.

Figure 2.11 illustrates the nodal estimation error field, E, and the obtained anisotropic mesh. Higher errors are associated to higher gradients, which will be enriched with more local nodes. Increasing the total number of nodes gives lower maximum values of the estimated error.



(a) $N = 3200$ (b) $N = 13000$ (c) $N = 35000$

Figure 2.11: Computed nodal estimated error field, E, and adapted anisotropic mesh, for three given number of nodes, N, of 3200, 13000 and 35000.

Figure 2.12 shows the solution of the interpolation, u_h, on these meshes, after the 15 performed iterations. One observes that it approximates better the initial image than the equivalent number of nodes isotropic mesh.

(a) $u_h, N = 3200$ (b) $u_h, N = 13000$ (c) $u_h, N = 35000$

Figure 2.12: Solution of the interpolation, u_h, for three adapted meshes with different increasing number of nodes, N, of 3200, 13000 and 35000.

It also shows the potential of anisotropic adaptation to reduce the stored information and perform image compression. As introduced in Section 2.3, let us build a new image, $\widehat{u_h}$, with the same number of nodes $\widehat{T}$ as the original image, but based on the result of the interpolation on a mesh with N nodes ($N < \widehat{T}$), with or without adaptation iterations. To analyze the quality of the results, we plot the MSE (Mean Square Error) as a function of density D. Figure 2.13 shows the influence of the density on the computed MSE for "Lena"'s image, using uniform and anisotropic meshes. We observe that, for the same density, the anisotropic adapted meshes provide less error than uniform ones, as expected, since nodal placement is, in this case, optimized.

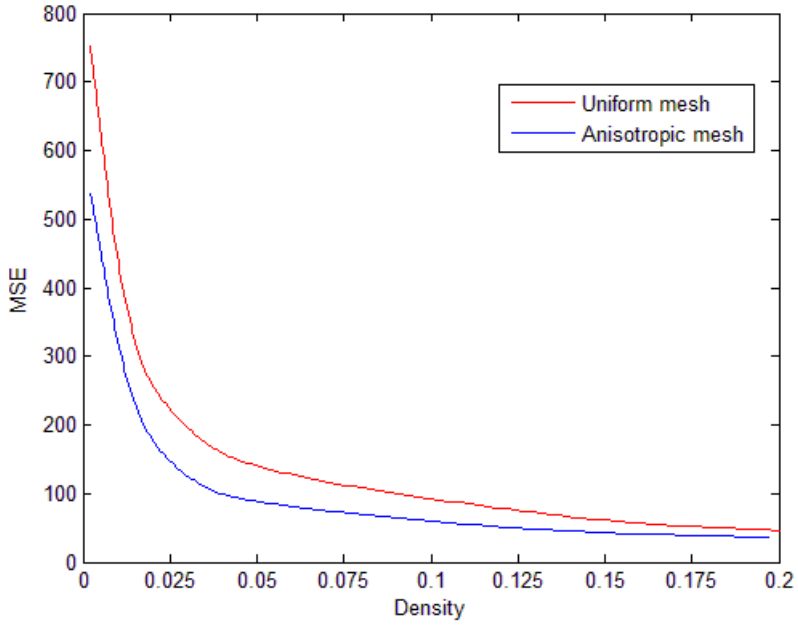


Figure 2.13: "Lena"'s image MSE as a function of the density, for uniform and anisotropic adapted meshes.

2.4.6.2 Application to 2D/3D head MRIs

The proposed methodology can be potentially used in medical applications. Figure 2.14(a) is a 2D grey-scale MRI of the human head given by the Optima* MR450w with GEM Suite MRI system [49]. It contains 490000 pixels ($\widehat{L} \times \widehat{H} = 700 \times 700$). This image has been firstly interpolated on an initial uniform mesh. The interpolation results, for different number of fixed nodes, is shown in Figure 2.14.

By increasing the number of nodes, the skull and the brain regions become better and brain wrinkles appear clearly. In fact, results for a mesh with 127518 nodes compares very well with the initial image, which has 490000 pixels. Furthermore, image interpolation coupled to the automatic anisotropic mesh adaptation procedure improves the results, using less nodes. Figure 2.15 illustrates the resulting interpolation of a 2D cut of the MRI head on an adapted anisotropic mesh, for different number of nodes. For runs on one core and different number of fixed nodes, the CPU times (s) are presented in Table 2.2.

N	3600	13000	32000	53000
Time (s)	50	107	202	331

Table 2.2: CPU time (s) for interpolation and mesh adaptation for different number of nodes and 15 mesh iterations, runs performed on one core.

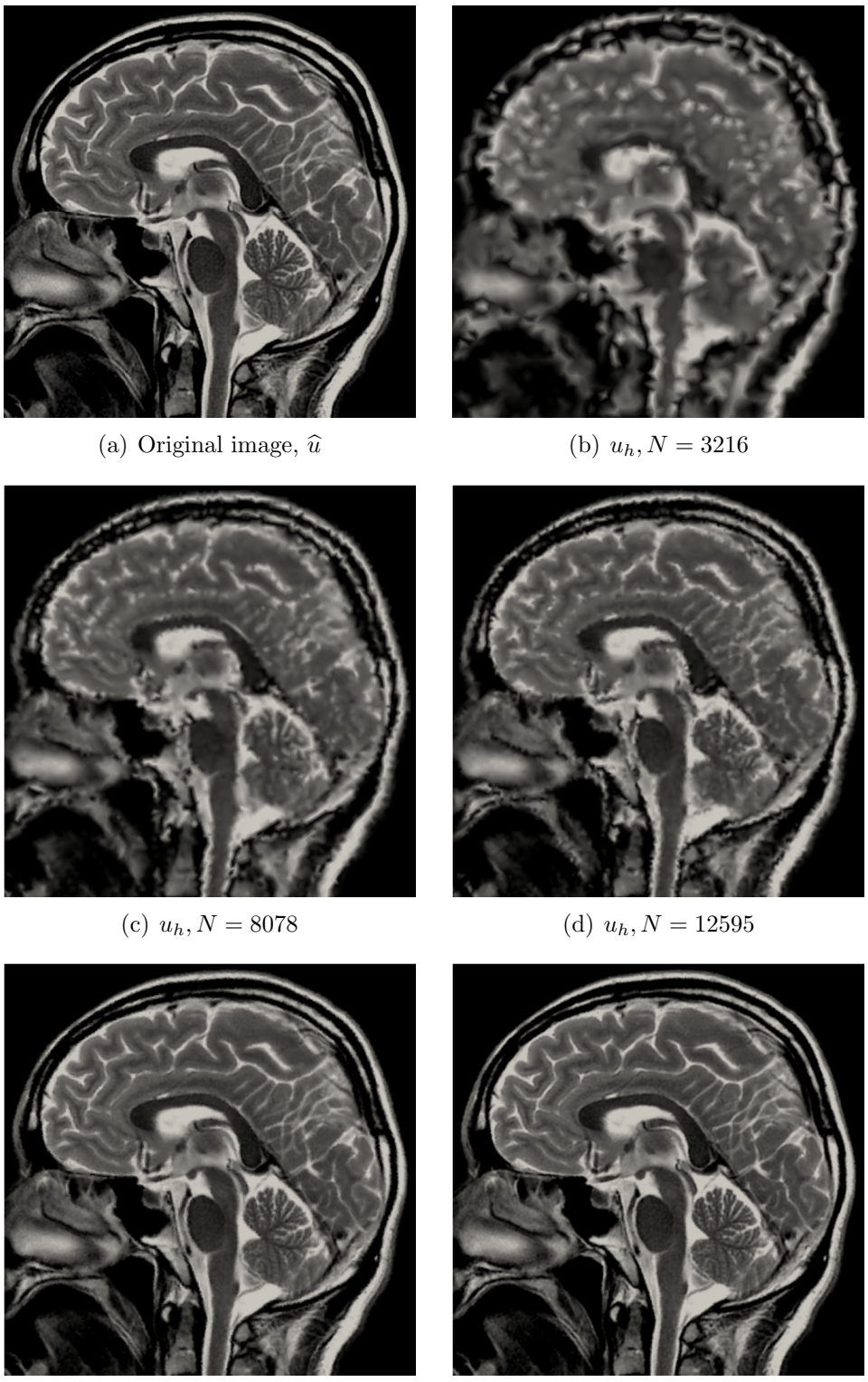
We observe that, using only 13000 nodes, the interpolation coupled to an adapted anisotropic mesh performs very well. In addition, for 53000 nodes, the final mesh clearly describes details like the brain wrinkles and other important features which may be used to improve final purpose computations.

The 3D case, presented in Figure 2.5(a) and interpolated in uniform meshes, has also been considered. The same algorithm was run on 4 cores, also imposing a larger number of nodes. Figure 2.16 shows the results obtained by coupling interpolation and adaptation in this case. 3D anisotropic mesh adaptation requires more iterations than 2D to attain the same mesh quality. CPU time (min), for different number of imposed nodes and 25 iterations is presented in Table 2.3.

N	100000	200000	300000	400000
Time (min)	66.5	115.5	189.9	227.7

Table 2.3: CPU time (min) for image interpolation and mesh adaptation in 3D, with different number of nodes and for 25 iterations, runs performed on 4 cores.

Quantitative comparison, is terms of MSE, is given in Figure 2.17, by plotting it as a function of the density D, for both the 2D and 3D cases. We observe that, for the same density, anisotropic adapted meshes provides less error than uniform ones, in particular for the 3D images, where mesh compression gains are more relevant.



(a) Original image, $\widehat{u}$

(b) $u_h, N = 3216$

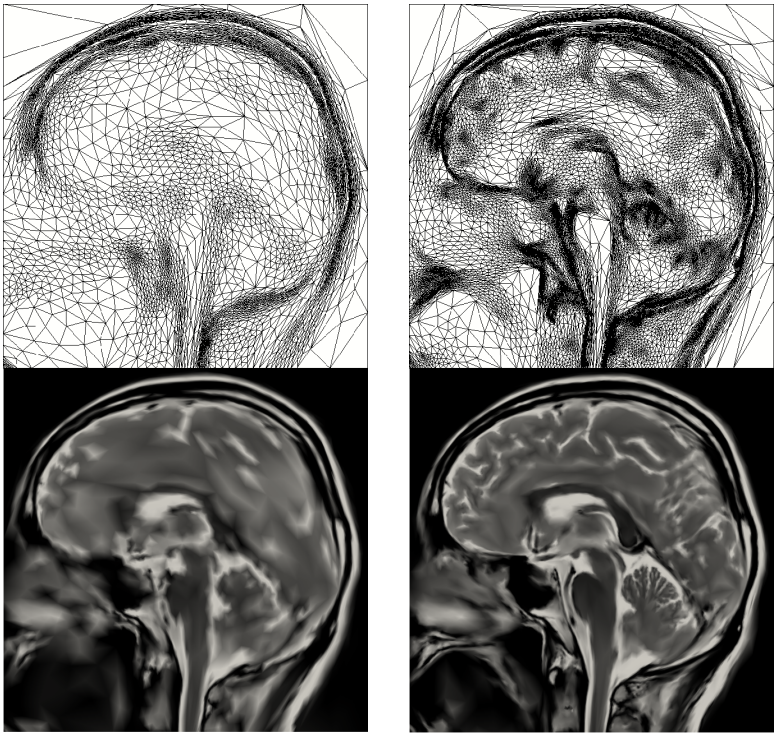
(c) $u_h, N = 8078$

(d) $u_h, N = 12595$

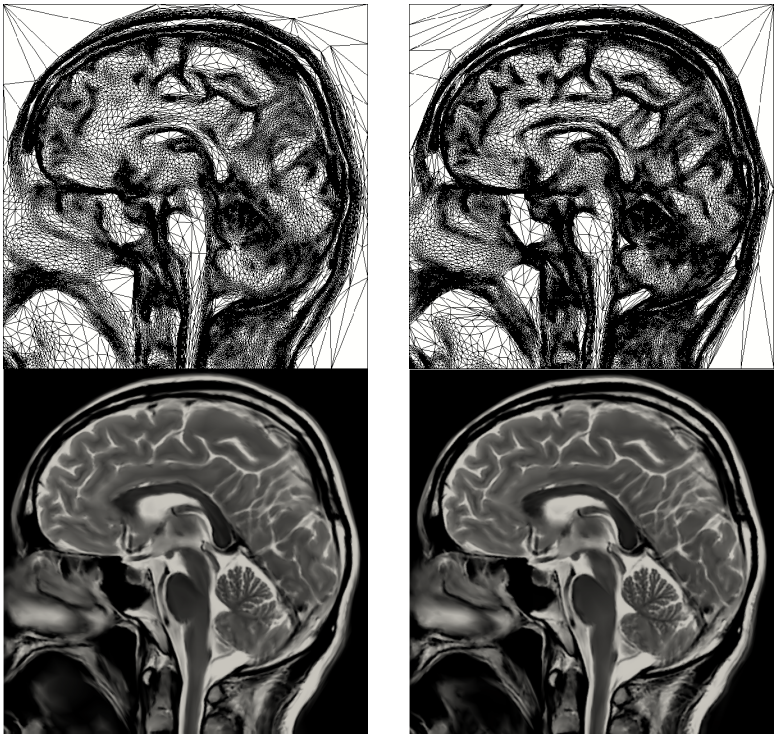
(e) $u_h, N = 50372$

(f) $u_h, N = 127518$

Figure 2.14: Original 2D MRI cut of the scan of the head, and application of the interpolation of the image in 2D uniform meshes with different number of nodes N.

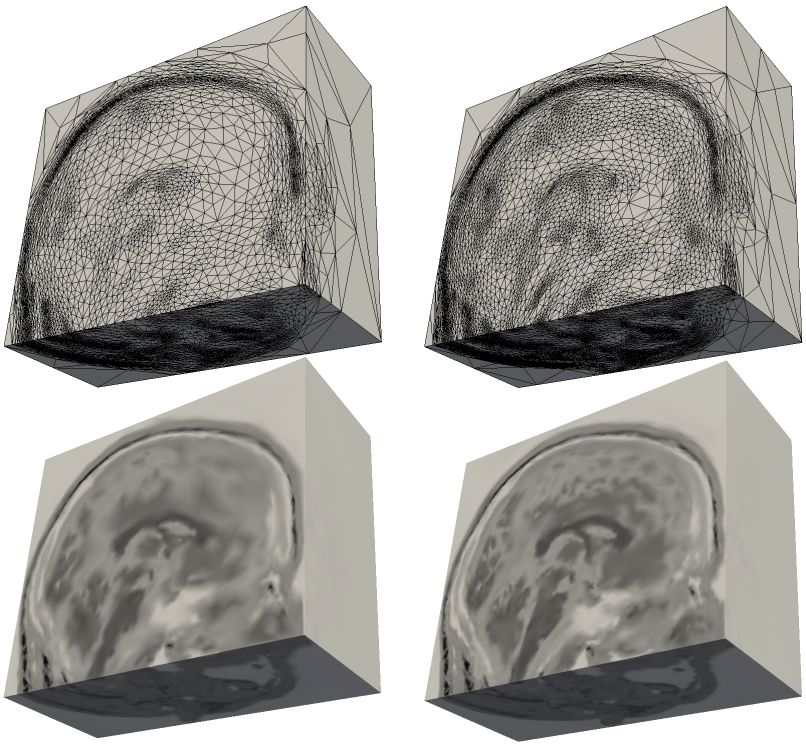


(a) $N = 3600$ (b) $N = 13000$

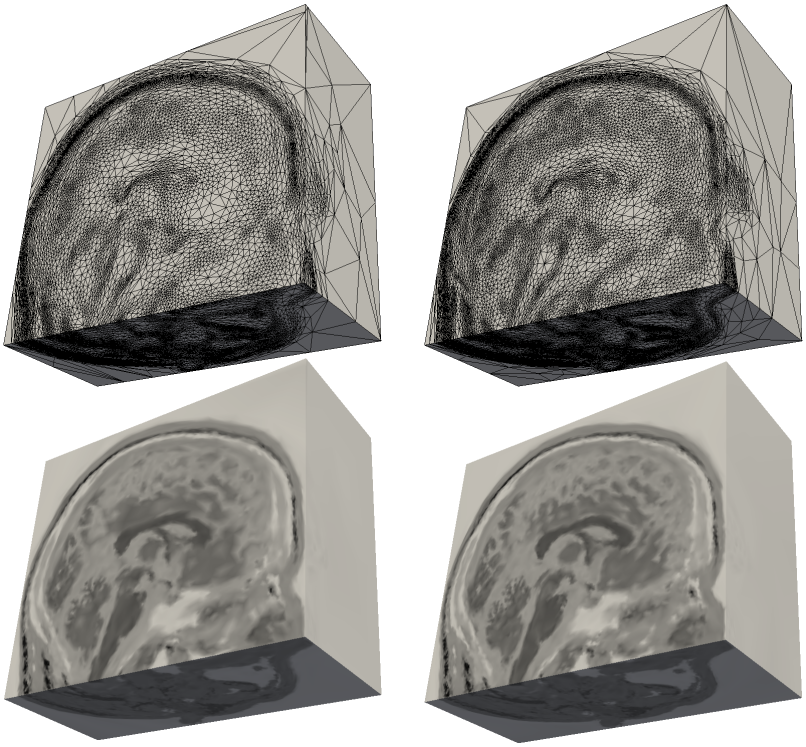


(c) $N = 32000$ (d) $N = 53000$

Figure 2.15: Final anisotropic mesh for different imposed number of nodes N and resulting interpolation for a 2D MRI scan of the human head.

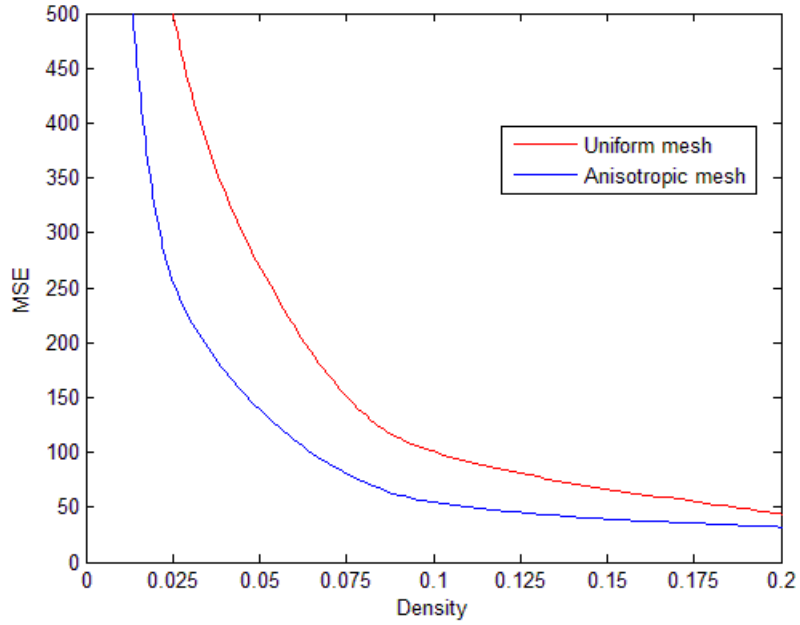


(a) $N = 100000$ (b) $N = 200000$

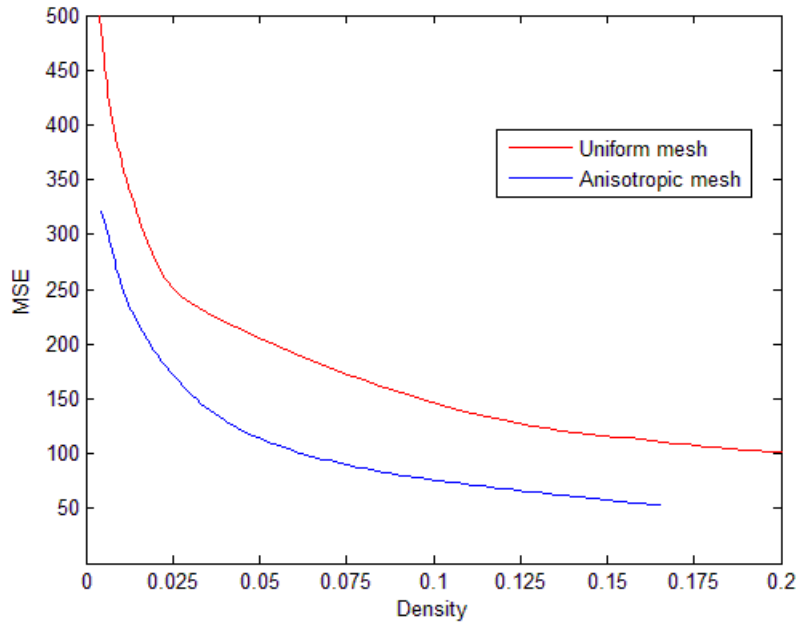


(c) $N = 300000$ (d) $N = 400000$

Figure 2.16: Final anisotropic mesh with different imposed number, for a nodes N and resulting interpolation for 3D a MRI scan of the human head.



(a) 2D MSE



(b) 3D MSE

Figure 2.17: Mean Square Error (MSE) on 2D and 3D MRIs of the human head as a function of the density for different sized uniform and anisotropic meshes.

2.4.6.3 Application to 2D color images

Figure 2.18 is a RGB color image of a cut on a steel polycrystal, $\widehat{u_{RGB}}$, 24-bits, and of dimension (1000×1000). During the cooling this type of material, orientation of crystallites may be rather random, presenting different directions, each represented by a different color.

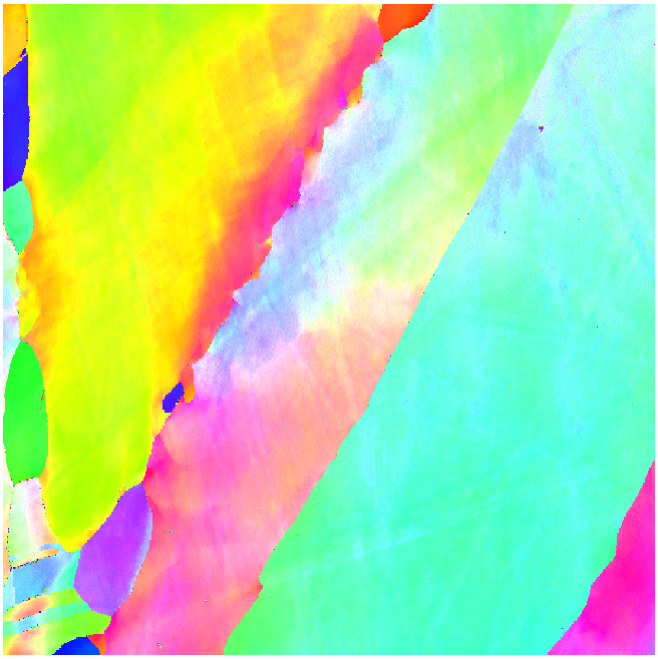
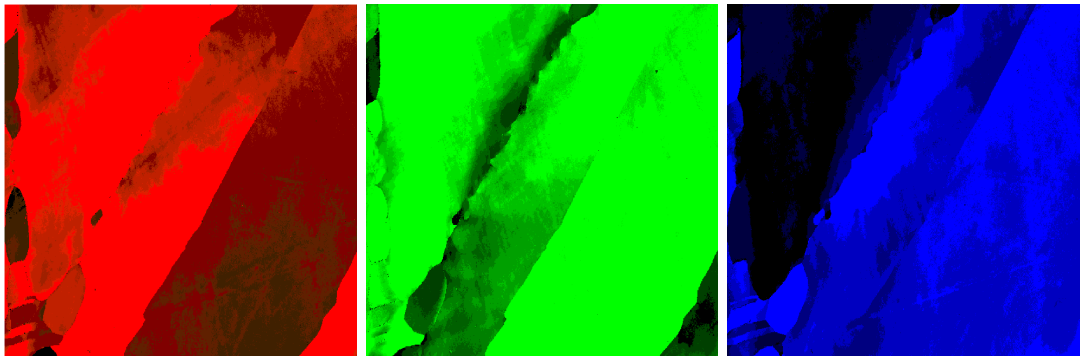


Figure 2.18: Color image of a steel polycristal, $\widehat{u_{RGB}}$.

This color image, $\widehat{u_{RGB}}$, may be decomposed into three sub-images, in three channels. Figures 2.19(a),(b) and (c) show the value of the Red, Green and Blue channels, $\widehat{u_{red}}$, $\widehat{u_{green}}$, $\widehat{u_{blue}}$, each channel varying from 0 to 255.

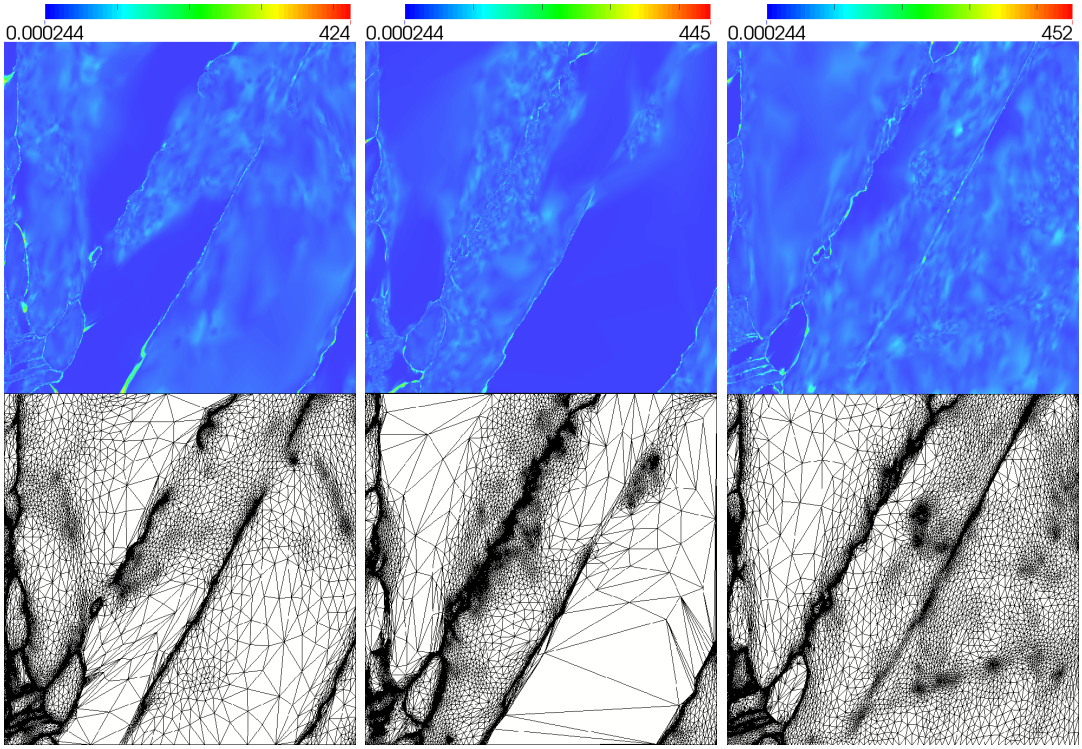


(a) Red channel $\widehat{u_{red}}$ (b) Green channel $\widehat{u_{green}}$ (c) Blue channel $\widehat{u_{blue}}$

Figure 2.19: Three channels, red, green and blue, result of the decomposition of the 24-bits steel polycrystal color image.

Firstly, we have used the immersed image method to adapt these three images in an independent way, fixing $N = 40000$. The computed estimated error and the

obtained adapted anisotropic mesh for this case are shown in Figures 2.20(a),(b) and (c). Each mesh is well adapted for the corresponding image, but none of them are well adapted for the original RGB image.



(a) Adapted R channel (b) Adapted G channel (c) Adapted B channel

Figure 2.20: Computed estimated error and adapted anisotropic mesh for each channel, fixing 40000 nodes for the adaptation procedure.

Therefore, we combine the values of the three channels to create the vector field $\vec{u} = \{u_{red}, u_{green}, u_{blue}\}$. As introduced in Section 2.3.4, the estimated error is computed for each component, generating the error vector, $\overrightarrow{e^{ij}} = \{e^{ij}_{red}, e^{ij}_{green}, e^{ij}_{blue}\}$. Then, the L_2 norm of this vector is determined and is the one used to construct the metric $\widetilde{\mathcal{M}}$. Figure 2.21 shows the computed nodal estimated error, $\|e^{ij}\|_{L_2}$ and the obtained anisotropic mesh. We observe that the final mesh is well adapted to all of these three channels.

2.5 Dynamic parallel adaptation

Nowadays, computer capabilities are guided, not by the improvement of the processors speed, but by generating multi-core systems. In this context, softwares necessarily need to be adapted to such techniques and should be fully parallelized. The principle of parallelization is to divide the whole problem into many small ones, which are solved at the same time on the different cores. Parallelization methods developed and employed for mesh adaptation and image immersion are described in the following.

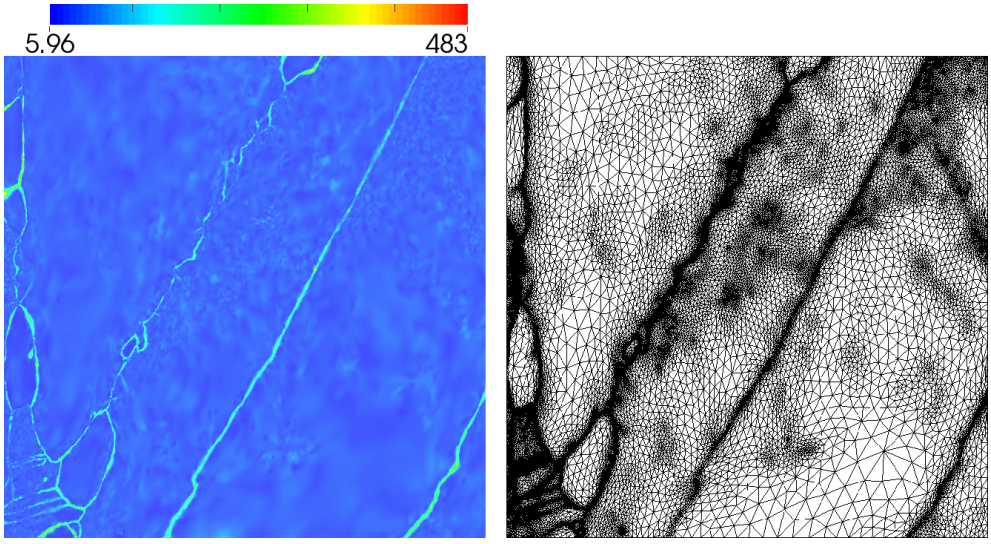


Figure 2.21: Computed estimated error and adapted anisotropic mesh for the vector compared of the three RGB channels, fixing $N = 40000$ nodes.

2.5.1 Parallel anisotropic mesh adaptation

The parallel method to handle mesh adaptation is introduced and has been fully detailed in [50].

To illustrate the implemented technique, we consider again the case of "Lena"'s image (Figure 2.4(a)), this time adapted using a different number of cores, from 1 to 4. Let us suppose a fixed number of nodes, $N = 100000$. After 15 iterations of immersion-adaptation, the mesh reaches convergence. The computing time as a function of the number of cores is shown in Table 2.4.

Number of cores	1	2	3	4
Time (s)	667	374	284	248
Memory (number of "total images") per increment	1	2	3	4

Table 2.4: The computing time and in-charged image for an increasing number of CPUs.

The whole image is charged by each core whereas the imposed 100000 nodes are equally distributed between the cores. Each core constructs the anisotropic mesh of its own region to adapt to the image information. This parallel method reduces the computing time, but also increases the memory storage requirement because of the need to give the whole image to all the cores.

2.5.2 Parallel image immersion and mesh adaptation

For larger images, the whole image load on each core is not an efficient option. To overcome to this problem, we have implemented an image partitioning technique, somehow related to the mesh partition so that the whole procedure is both time and memory performant.

To illustrate our approach, let us cut the whole Lena's into equally dimensioned sub-images. For example, we consider the 2×2 case with 4 sub-images ($4 \times (256 \times 256)$), the 4×4 case with 16 sub-images ($16 \times (128 \times 128)$), or the 8×8 case with 64 sub-images ($64 \times (64 \times 64)$), as seen in Figure 2.22.

(a) 2×2 sub-images (b) 4×4 sub-images (c) 8×8 sub-images

Figure 2.22: Lena's image cut into 4, 16, 64 sub-images.

For this example, we suppose that the mesh will be generated using 4 cores, being each sub-domain schematically represented in Figure 2.23(a). The first core manages the red region, whereas the other cores (2, 3, 4) take into their charge the remaining regions, which is illustrated in Figure 2.23(b). Since only the red region is treated by the first core, information of the other parts is not necessary. Each core will need to load the image information stored in the bounding box corresponding to its color (Figures 2.23(b) and (c)).

Let us suppose the core where the image has been partitioned in 4×4 sub-images (Figure 2.23(d)). Each core only charges the overlapping sub-images within its bounding box reducing the memory required. At the end, 36 sub-images are charged in this case, but with a memory reduction of 175%, when compared with loading the whole image in each core.

The implemented overall algorithm is given below.

Input: Sub-images, initial mesh $\mathcal{H}$ and given number of nodes N
Output: Adapted anisotropic mesh $\widetilde{\mathcal{H}}$ with N nodes, interpolation solution u_h on this mesh.

1 **while** *the adapted anisotropic mesh is not achieved* **do**
2 Partition the current mesh for the given number of cores and provide bounding boxes.
3 Select and charge the overlapping sub-images on the boundary box of each core.
4 Interpolate the sub-images on the current mesh.
5 Generate the optimal mesh $\widetilde{\mathcal{H}}$ using parallel anisotropic mesh adaptation.
6 Update new mesh.

Since the charged domain per core may be modified after each iteration (when mesh partition charges), it is necessary to recompute the bounding box of each mesh

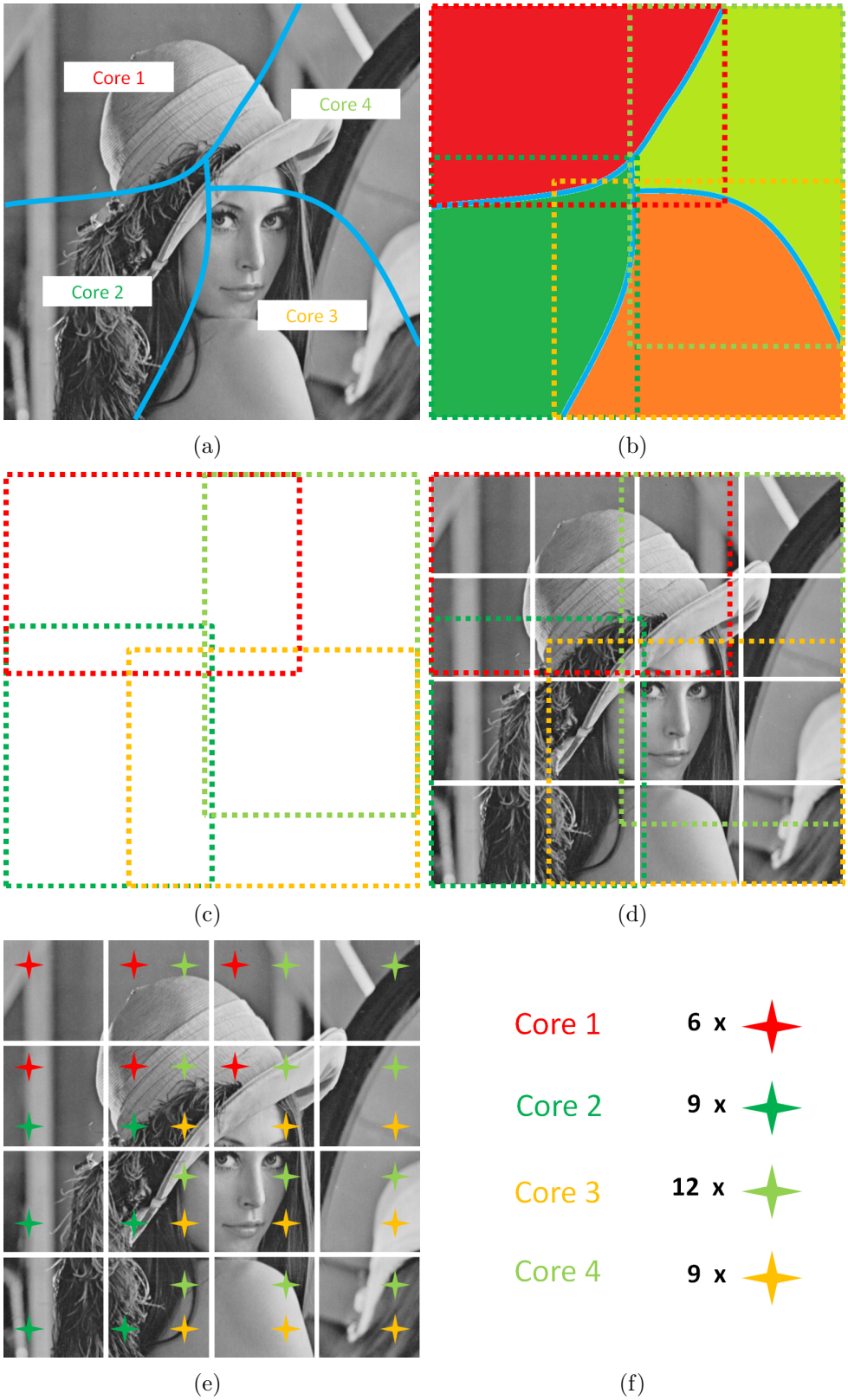


Figure 2.23: Illustration of the correspondence between image and mesh partitions, and on which image sub-domains are given to each mesh sub-domain.

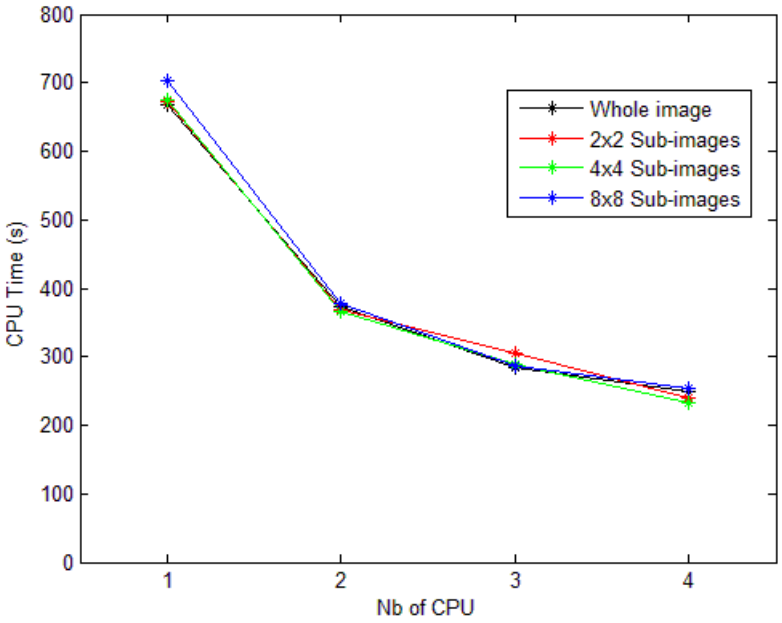
sub-domain and recharge the sub-images. As previously presented, let us consider the image-immersion with parallel partitioning of mesh and image applied to Lena's case, supposing that the computation is done in one to four cores, but that the image may be divided in 2×2, 4×4 and 8×8 sub-images. For each case, after 15 iterations, we have counted the total number of charged sub-images, and computed the equivalent number of "total images" charged per iteration. Results are given in Table 2.5.

2×2 case				
Nb of cores	1	2	3	4
CPU Time (s)	673	368	306	239
Nb of charged sub-images per iteration	4	8	10.5	11.8
Memory (number of "total image") per iteration	1	2	2.63	2.96
4×4 case				
Nb of cores	1	2	3	4
CPU Time (s)	676	366	288	233
Nb of charged sub-images per iteration	16	30.4	35.3	37.4
Memory (number of "total image") per iteration	1	1.90	2.20	2.33
8×8 case				
Nb of cores	1	2	3	4
CPU Time (s)	702	377	287	253
Nb of charged sub-images per iteration	64	116.8	130.1	135.0
Memory (number of "total image") per iteration	1	1.82	2.03	2.11

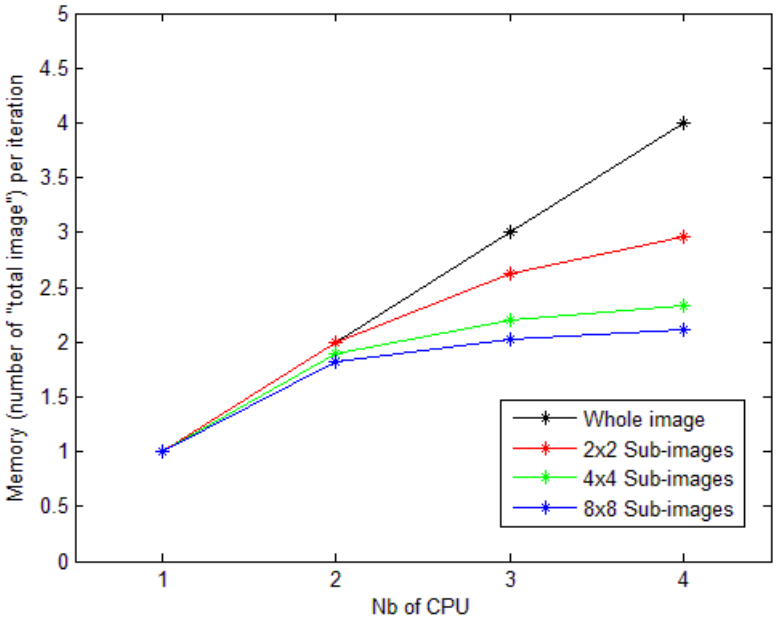
Table 2.5: The CPU time (s), the number of charged sub-images and number of equivalent charged total images per iteration for a total number of $CPUs$, for, 2×2, 4×4 and 8×8 sub-image case.

Figure 2.24 plots the computation time and the total memory required, per increment, for the whole image, in the 2×2, 4×4 and 8×8 sub-images cases, for different number of cores.

Even if the computational time is rather equivalent for all the cases, charging a higher number of sub-images leads to less memory requirements, as expected.



(a) Computational time



(b) Charged memory per iteration

Figure 2.24: Comparison of the computational time and of the charged memory per iteration when dealing with the whole image, 2×2, 4×4 and 8×8 sub-images, for different number of cores.

2.6 Conclusion

In this chapter, a new method to directly construct anisotropic mesh based on image data has been presented. Image interpolation on a mesh may directly transport the information and provides discretization values on this mesh. An error estimation operator allows the computation of the metric field, given to a topological optimization mesh generator. This last iteratively improves the mesh to be adapted to the image data, and its interpolation results approximate the image. This method can handle grey-scale images but also color ones, by performing multi-component mesh adaptation.

Mesh adaptation has been formerly parallelized, but some issues concerning image arise. The major one is the huge memory storage requirement for larger images, if the image is given to each core. To overcome to this drawback, image parallelization has been proposed. The original image is partitioned into several sub-domains and coupled to mesh partition, accelerating the adaptation procedure and reducing the memory storage.

2.7 Résumé français

Dans ce chapitre, nous avons présenté une nouvelle méthode de construction de maillage basée sur la simple donnée d'une image. L'interpolation de l'image sur un maillage permet d'importer directement les informations de l'image et fourni directement des valeurs de discrétisation. Un estimateur d'erreur permet la construction d'un champ de métrique et d'un maillage optimal. Cette méthode ne s'applique pas seulement aux images en noir et blanc, mais aussi aux images couleur en utilisant une adaptation de maillage à multi-composantes.

La méthode implementée pose néanmoins certains problèmes. Le plus contraignant est celui de la mémoire pour les grandes images. La parallélisation de l'image permet donc d'écarter ce problème. L'image et le maillage sont donc partitionnés en plusieurs sous-domaines, accélérant l'adaptation et rduisant le stockage mémoire.

Chapter 3

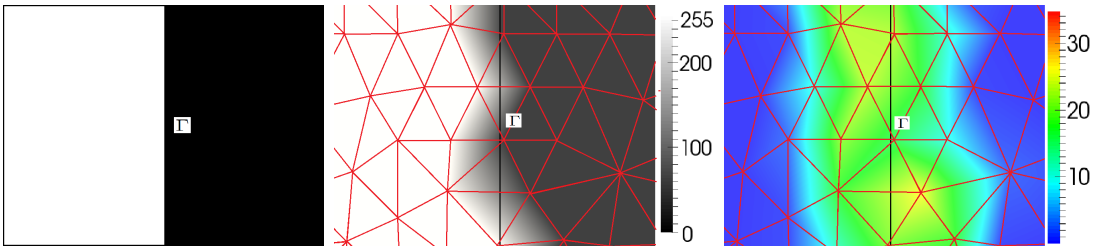
Redistancing coupled to anisotropic mesh adaptation

Contents

3.1 Introduction

For numerical simulation reasons, it is often necessary to distinguish different objects belonging to the image. For example, when the image presents zones with different physical properties corresponding to different values of the pixel/voxel. Sometimes, considering the limit of image acquisition techniques, this is difficult to identify in the initial image, so that image processing is necessary and may provide distinguished and homogenous zones. For that, segmentation algorithms have been proposed and implemented like, for example, the thresholding method, the edge detection method or the active contour model widely described in the previous chapter and in references [51, 7, 6, 9, 11, 13, 52].

Basically, when using image processing the original image $\widehat{u}$ becomes the segmented one $\widehat{u_{seg}}$, with identified segmented zones. To illustrate this with a simple example, let us consider a two phase system where the corresponding original image has been segmented, providing $\widehat{u_{seg}}$ (Figure 3.1(a)), supposing the pixel's value equal to 255 inside one phase and 0 elsewhere. If the segmented image $\widehat{u_{seg}}$ is interpolated on an existing uniform mesh, giving u_h, one may observe than coupling mesh adaptation directly on this u_{seg} using the previous presented methodologies may not provide an acceptable anisotropic mesh, since the pixel value is discontinuous at the boundary of the segmented region. Figure 3.1(b)(c) shows the interpolation solution and the computed nodal estimated error E around the interface between the two phases, drawn on the initial uniform mesh. We find that this error is high around this interface and zero far away.



(a) $\widehat{u_{seg}}$ (b) Interpolation solution, u_h (c) Nodal estimated error, E

Figure 3.1: Illustration of the segmented image interpolation on an initial uniform mesh.

As previously presented, new nodes will be created in the region with high error, intending to reduce and equi-distribute it within the mesh. However, this result is contrary to our expectations, where the new interpolation solution is computed as previously (Figure 3.1(b)). For binary images, the nodal estimation error E is large around the interface, since this type of segmentation introduces a discontinuity.

Then, an infinite number of nodes at the interface is required during the anisotropic mesh adaptation procedure to be able to reach the target error.

For computational reasons [38, 53, 54, 55], mesh with such a density of nodes at the interface will not be helpful for numerical simulation. The latter requires smooth continuous functions around the phase's boundary, which will be used to define physical parameters distribution on the computational domain and to improve the mesh adaptation procedure like, for example, in multiphase computational fluid

dynamics simulations. Indeed, our primary goal is thus now to build a regularized function around an image segmented object.

3.2 Construction of a regularized function using a redistancing method

3.2.1 Level-set approach

To build a smooth phase function, the first candidate would be the level-set function. In the last decades, many image segmentation techniques [11, 13] and simulation methods were based on implicit functions (like the level-set one). Level-set methods have been initially developed by Osher [12] and are now widely used for the analysis of surfaces and shapes, especially in multiphase representations, as well as for capturing interfaces of different objects, between others. Let us note the level-set function as u_d, which may be the signed distance function to the boundary of the object ω. Γ is this boundary of ω, which is given by the iso-zero value of u_d (as seen in Figure 3.2), such that:

$$u_d = \begin{cases} \mathrm{d}(x,\Gamma) & \text{if } x \in \omega \\ 0 & \text{if } x \in \partial\omega \\ -\mathrm{d}(x,\Gamma) & \text{if } x \in \Omega \setminus \omega \end{cases} \tag{3.1}$$

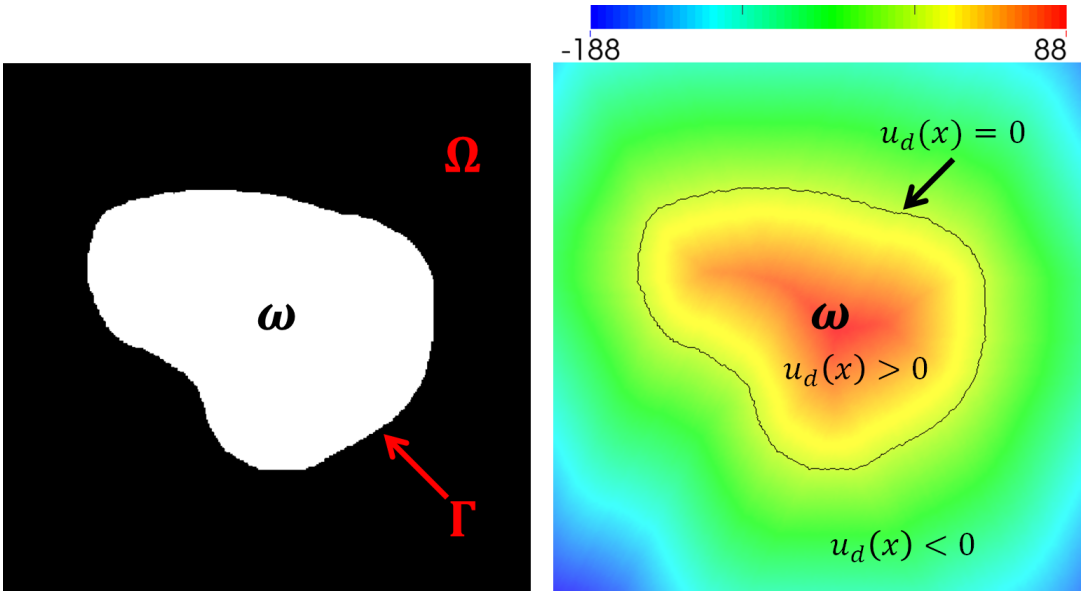


Figure 3.2: Object ω and the signed distance function to its boundary, u_d.

3.2.2 Redistancing a modified level-set function

Level-set functions for capturing moving interfaces have been developed for many years. The interface is identified as the zero-level and the level-set function must be

as regular as possible to keep a good numerical accuracy. Level-set methods have been improved through redistancing [56, 57] since, for non-homogeneous velocity fields, metric properties of level-sets are not guaranteed. By fixing the zero-level, Hamilton-Jacobi properties may be used to recreate level-sets.

Let us recall the classical level-set redistancing method. Firstly, we intend to rebuild a signed distance function, u_d, only based on the knowledge of its sign, $S(u_d^0)$. The sign function is determined by the phase's placement in the computational domain and only depends on the zero-level of the initial function, $u_d^0(x \in \partial\Omega)$. Then,

$$S(u_d^0) = \begin{cases} 1 & \text{if } x \in \omega \\ 0 & \text{if } x \in \partial\omega \\ -1 & \text{if } x \in \Omega \setminus \omega \end{cases} \tag{3.2}$$

The main proprerty of the signed distance is that $\|\nabla u_d\|_2 = 1$. Then, one may solve a Hamilton-Jacobi equation [56, 57] to obtain a regular signed distance function, when this is not the case, through

$$\begin{cases} \dfrac{\partial u_d}{\partial \tau} = S(u_d^0)(1 - ||\nabla u_d||_2) \\ u_d(x, \tau = 0) = u_d^0(x) \end{cases} \tag{3.3}$$

The final redistanced function u_d^τ is a signed distance and has the same boundary Γ as the initial function u_d^0. This function is also differentiable everywhere in the domain and its gradient satisfies the Eikonal equation $\|\nabla u_d\| = 1$. However, since its gradient is of constant value, it will not be useful to enrich the mesh around the interface. Therefore, one other solution is to use an hyperbolic tangent function [37, 58] of the signed distance, with thickness ε:

$$u_\varepsilon(\varepsilon, u_d) = \varepsilon \tanh(\frac{u_d(x, \Gamma)}{\varepsilon}) \tag{3.4}$$

We prefer to use $u_\varepsilon(\varepsilon, u_d)$ instead of the level-set function, which is also smooth and regularized around an object of interest. There are two important advantages on using it:

- the second order gradient of the hyperbolic tangent function is the one directly used to estimate the error in anisotropic mesh adaption, well posed for that;
- the hyperbolic tangent function may be directly applied in mixture laws [58] to define the different physical parameters in the whole computing domain.

ε is a parameter that defines an "interface thickness", since u_ε varies between -ε and ε over a thickness that is, approximately, 3ε. The gradient's norm of u_ε is

$$||\nabla u_\varepsilon||_2 = 1 - (\frac{u_\varepsilon}{\varepsilon})^2 = g(u_\varepsilon) \tag{3.5}$$

Inspired by the redistancing methodology [56, 57], this scalar field $u_\varepsilon^\tau(\varepsilon, S, \tau) = u_\varepsilon(\varepsilon, u_d)$ is also the solution of the following Hamilton-Jacobi or Eikonal equation:

$$\begin{cases} \dfrac{\partial u_\varepsilon}{\partial \tau} + S\left[||\nabla u_\varepsilon||_2 - g(u_\varepsilon)\right] = 0 \\ u_\varepsilon^0 = u_\varepsilon(\varepsilon, S, \tau = 0) \end{cases} \tag{3.6}$$

where S is the sign function which depends, for our problems, on the initial image and on the pixel/voxel values defining the boundary of the object for which we wish to build u_ε. To have it, we can do a rather simple threshold:

$$S(\mathbf{X}^i) = \begin{cases} 1 & \text{if } \widehat{u}(Pixel^k/Voxel^k) > \text{ threshold value} \\ 0 & \text{if } \widehat{u}(Pixel^k/Voxel^k) = \text{ threshold value} \\ -1 & \text{if } \widehat{u}(Pixel^k/Voxel^k) < \text{ threshold value} \end{cases} \tag{3.7}$$

For $\forall\ i = 1, \ldots, N$, at the mesh node $\mathbf{X}^i$, the sign function S is the key of the redistancing method and plays two important roles:

- it controls the direction of redistancing, starting around the zero-level, towards the exterior;
- it is a corrector. If the sign of the solution of Equation (3.6) is not the same as the sign computed from the image, a correction of u_ε^τ may be performed after each resolution, by considering the following step:

$$u_\varepsilon^\tau(\mathbf{X}^i) = S(\mathbf{X}^i)\left|u_\varepsilon^\tau(\mathbf{X}^i)\right| \tag{3.8}$$

Finally, Equation (3.6) may also be rewritten as an equation of the hyperbolic type

$$\frac{\partial u_\varepsilon}{\partial \tau} + v_r \cdot \nabla u_\varepsilon = S(\mathbf{X}^i) g(u_\varepsilon) \tag{3.9}$$

where v_r is the redistancing velocity, defined as

$$v_r = S(\mathbf{X}^i)\frac{\nabla u_\varepsilon}{||\nabla u_\varepsilon||_2} \tag{3.10}$$

3.2.3 Stabilized finite-element resolution

The redistancing method is stable, the equations are of a well defined type and present no particular problems in two or three dimensions. They may be solved using the finite element method, as described in the following.

3.2.3.1 Variational formulation

Let us recall the variational formulation of the given problem, and the application of the standard Galerkin finite element method. Some important notations will be firstly presented .

The Sobolev space of functions $H^1(\Omega)$ has square integrable first order derivatives in L^2 and is a Hilbert vector space. We may define

$$\begin{cases} \mathcal{V} = H^1(\Omega) = \left\{ u \in L^2(\Omega), \|\nabla u\| \in L^2(\Omega) \right\} \\ \mathcal{Q} = L^2(\Omega) = \left\{ u(x) | \int_\Omega \left| u^2 \right| dx \right\} \\ \mathcal{V}^0 = H_0^1(\Omega) = \left\{ u \in H^1(\Omega) | u = 0, \text{ at } \partial\Omega \right\} \end{cases} \tag{3.11}$$

The subspace $H_0^1(\Omega) \subset H^1(\Omega)$ gives the set of functions vanishing on the boundary of Ω. In $L^2(\Omega)$, the scalar product $(,)$ is defined as

$$(u, v) = \int_\Omega |uv| \, dV \tag{3.12}$$

The pure convective Equation (3.9) is written by considering an appropriate function $w \in \mathcal{W}^0 = H_0^1(\Omega)$. The integral over the whole computed domain may be written as:

$$\int_\Omega \frac{\partial u_\varepsilon}{\partial \tau} \cdot w + \int_\Omega \nabla u_\varepsilon \cdot v_r \cdot w = \int_\Omega F \cdot w \quad \forall w \in \mathcal{W}^0 \tag{3.13}$$

or, in a scalar product $(,)$ notation, as:

$$(\frac{\partial u_\varepsilon}{\partial \tau}, w) + (v_r \nabla u_\varepsilon, w) = (F, w) \quad \forall w \in \mathcal{W}^0 \tag{3.14}$$

where $F = S(\mathbf{X}^i) g(u_\varepsilon)$. To build the spatial discretization for the finite element formulation, the entire computational domain Ω is discretized into simplex elements, K. Using the above notations, $\mathcal{V} \subset H^1(\Omega)$ and $\mathcal{V}^0 \subset H_0^1(\Omega)$, the discrete spaces are designated $\mathcal{V}_h = H^{1h}(\Omega)$ and $\mathcal{V}_h^0 = H_0^{1h}(\Omega)$, and are spaces of piecewise linear functions. The smaller the mesh size of the discretization, the more accurate the approximation of the functional spaces:

$$\lim_{h \to 0} \mathcal{V}_h = \mathcal{V} \text{ and } \lim_{h \to 0} \mathcal{V}_h^0 = \mathcal{V}^0 \tag{3.15}$$

The discretized formulation is as follows:

$$(\frac{\partial u_{\varepsilon h}}{\partial \tau}, w_h) + \underbrace{(v_r \nabla u_{\varepsilon h}, w_h)}_{\mathcal{B}(u_{\varepsilon h}, w_h)} = \underbrace{(F_h, w_h)}_{\mathcal{F}(w_h)} \quad \forall w_h \in \mathcal{W}_h^0 \tag{3.16}$$

Finally, the problem to be solved is a system of first order differential equations:

$$\mathbf{A} \frac{\partial \mathbf{U}}{\partial \tau} + \mathbf{B} \cdot \mathbf{U} = \mathbf{F} \tag{3.17}$$

3.2.3.2 Time integration scheme

The temporal domain, $\Delta\tau \in [0, \tau]$, is separated into I regular intervals, so that:

$$\Delta\tau = \frac{\tau}{I} \tag{3.18}$$

where $\Delta\tau$ is a fictitious time step and the number of performed iterations is $n = 1, \ldots, I$. Using a general finite difference scheme, the temporal discretization of the convection problem, Equation (3.17), is as follows:

$$\mathbf{A}\frac{\mathbf{U}^I - \mathbf{U}^{I-1}}{\Delta\tau} + \mathbf{B}(\theta\mathbf{U}^I + (1-\theta)\mathbf{U}^{I-1}) = \mathbf{F} \tag{3.19}$$

where $0 \leq \theta \leq 1$. For $\theta = 0$, we have the forward Euler scheme, if $\theta = 0.5$ the Crank-Nicholson one and when $\theta = 1$ the backward Euler scheme. The used scheme here has $\theta = 0.5$, arising the following semi implicit formulation:

$$\left(\frac{u^I_{\varepsilon h}}{\Delta\tau}, w_h\right) + \frac{1}{2}\left(\nabla u^I_{\varepsilon h} \cdot v_r, w_h\right) = \left(\frac{u^{I-1}_{\varepsilon h}}{\Delta\tau}, w_h\right) - \frac{1}{2}\left(\nabla u^{I-1}_{\varepsilon h} \cdot v_r, w_h\right) + (F_h, w_h) \quad \forall w_h \in \mathcal{W}^0_h \tag{3.20}$$

3.2.3.3 Stabilized methods

Classical Galerkin formulations are often not suitable for convection dominated problems, when the Péclet number is $Pe >> 1$ and non-physical oscillations appear. Stabilization methods usually used are the $SUPG$ (Streamline Upwind Petrov-Galerkin), by adding weighted residual terms [32, 33], or RFB (Residual-Free Bubbles) [34, 35], which enriches the space of the solutions with a space of bubble functions, defined at the element level.

3.2.3.3.1 Streamline Upwind Petrov-Galekin method

The $SUPG$ technique was firstly proposed by [32, 33]. This method has been developed from the standard Galerkin one, to avoid the oscillations, by introducing a new additional weighting term, $\tau_K^{SUPG}\mathbf{v} \cdot \nabla w_h$ to the standard Galerkin weighting function w_h, in the upwind direction. The modification of the weighting function is:

$$\widetilde{w_h} = w_h + \tau_K^{SUPG}\mathbf{v} \cdot \nabla w_h \tag{3.21}$$

The basic idea is that there is more weighting in the upstream direction and less in the downstream one, as illustrated in Figure 3.3.

The stabilization parameter τ_K^{SUPG} (constant per element) depends on the mesh size and on the norm of the velocity parameter. In [32], it was defined as:

$$\tau_K^{SUPG} = \frac{1}{2}\frac{h_K}{|\mathbf{v}_K|}\max\{0, (1 - \frac{1}{Pe_K})\} \tag{3.22}$$

where $\mathbf{v}_K$ is the constant velocity field on element K, given by the average nodal velocity on element K, and the Péclet number, Pe_K.

The new variational formulation becomes:

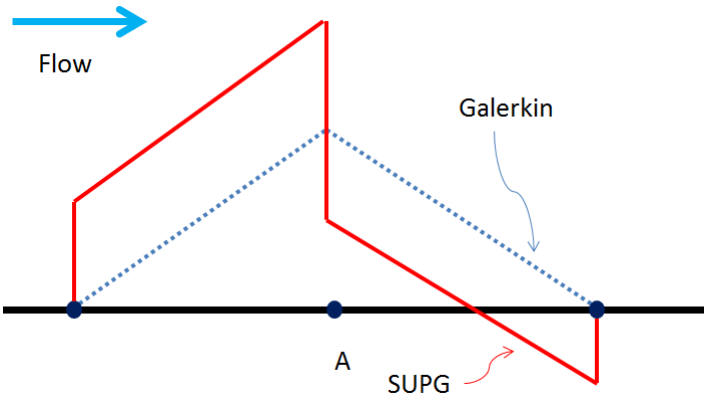


Figure 3.3: Ilustration and comparison between the Streamline Upwind/Petrov-Galerkin function and the Galerkin weighting one, for node **A** in a 1D mesh [32].

$$\underbrace{\left(\tfrac{\partial u_{\varepsilon h}}{\partial \tau}, w_h\right) + \sum_{K\in\mathcal{K}} \tau_K^{SUPG}\left(\tfrac{\partial u}{\partial \tau}, \mathbf{v}\cdot\nabla w_h\right)_K + \left(v_r \nabla u_{\varepsilon h}, w_h\right) + \sum_{K\in\mathcal{K}} \tau_K^{SUPG}\left(v_r\nabla u_{\varepsilon h}, \mathbf{v}\cdot\nabla w_h\right)_K}_{\mathcal{B}_\tau(u_{\varepsilon h}, w_h)}$$

$$= \underbrace{\left(F_h, w_h\right) + \sum_{K\in\mathcal{K}} \tau_K^{SUPG}\left(F_h, \mathbf{v}\cdot\nabla w_h\right)_K}_{\mathcal{F}_\tau(w_h)} \quad \forall w_h \in \mathcal{W}_h^0 \tag{3.23}$$

Comparing Equations (3.16) and (3.23), we observe that the $SUPG$ formulation adds a local diffusion along the stream direction. Finally, Equation (3.23) can be rewritten, like done for Equation (3.17), as a first order system of differential equations:

$$\mathbf{A}_\tau \frac{\partial \mathbf{U}}{\partial \tau} + \mathbf{B}_\tau \cdot \mathbf{U} = \mathbf{F}_\tau \tag{3.24}$$

Temporal discretization may then be performed on this system, like for the standard Galerkin formulation. Other expressions for the stabilization parameter τ_K^{SUPG} were also proposed in [33, 59, 60, 61]. The major drawback of the $SUPG$ stabilized method, is then the optimal choice of this parameter.

3.2.3.3.2 Residual-Free Bubble method

The basic principle of $SUPG$ methods is the addition of a stabilization parameter. However, there is one other way to handle stabilization by enriching directly the space functional, as is done in the RFB (Residual-Free Bubbles) method. This approach has been firstly proposed by [35] and developed in [62]. In this case, the space $\mathcal{V}_{RFB}$ is constructed by enriching the old space $\mathcal{V}_h$ with a bubble one, $\mathcal{V}_b$, so that:

$$\mathcal{V}_{RFB} = \mathcal{V}_h \oplus \mathcal{V}_b \tag{3.25}$$

where

$$\mathcal{V}_b = \bigoplus_{K \in \mathcal{K}} H_0^1(K) \tag{3.26}$$

Let b_K be a bubble function defined in element K, as seen in Figure 3.4, where the element is split in three sub-triangles or four sub-tetrahedra. The basic idea is then to enrich it with continuous piecewise linear polynomials. For example, in our equation, $\widetilde{u_\varepsilon} \in \mathcal{V}_{RFB}$ can be decomposed into the linear part $u_{\varepsilon h} \in \mathcal{V}_h$ and into a bubble b_K, with $u_{\varepsilon b} \in \mathcal{V}_b$:

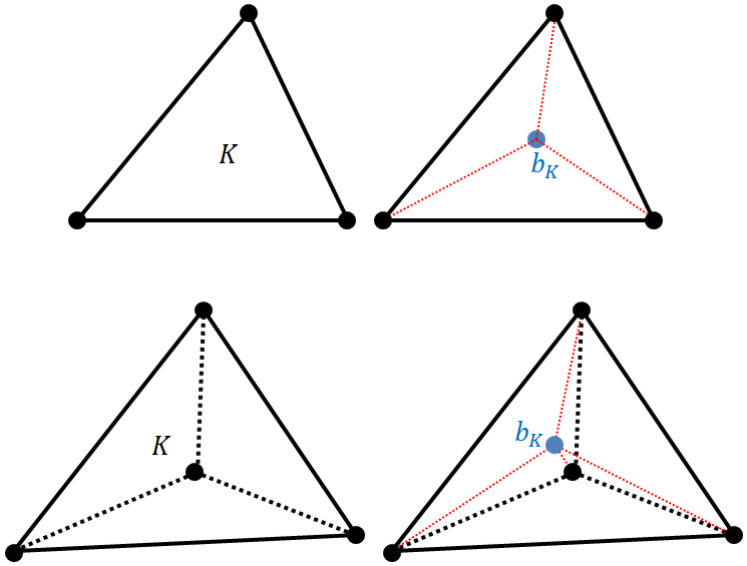


Figure 3.4: Triangle and tetrahedron elements, with the representation of the sub-domains and "bubbles".

$$\widetilde{u_\varepsilon} = u_{\varepsilon h} + u_{\varepsilon b} \quad \forall \widetilde{u_\varepsilon} \in \mathcal{V}_{RFB} = \mathcal{V}_h \oplus \mathcal{V}_b \tag{3.27}$$

where

$$u_{\varepsilon b} = \sum_{K \in \mathcal{K}} b_K \cdot u_{\varepsilon b_K} \tag{3.28}$$

More details about bubble functions are given and discussed in [63].

In the same manner, the space of test functions w_h can be split in two parts: $\widetilde{w} = w_h + w_b$, with the linear part $w_h \in \mathcal{W}_h^0$ and the bubble function $w_b \in \mathcal{W}_b^0$, so that

$$\widetilde{w} = w_h + w_b, \; \forall w_h \in \mathcal{W}_{RFB} = \mathcal{W}_h^0 \oplus \mathcal{W}_b^0 \tag{3.29}$$

Indeed, Equation (3.16) can be decomposed in two parts, the large-scale equation and the fine-scale one. The problem becomes: find $u_{\varepsilon h} \in \mathcal{V}_{RFB}$, such that

$$\begin{cases} \underbrace{(\dfrac{\partial(u_{\varepsilon h}+u_{\varepsilon b})}{\partial\tau},w_h)+(v_r\nabla(u_{\varepsilon h}+u_{\varepsilon b}),w_h)=(F_h,w_h)}_{\text{Large-scale equation}} & \forall w_h\in\mathcal{W}_h^0 \\ \underbrace{(\dfrac{\partial(u_{\varepsilon h}+u_{\varepsilon b})}{\partial\tau},w_b)+(v_r\nabla(u_{\varepsilon h}+u_{\varepsilon b}),w_b)=(F_h,w_b)}_{\text{Fine-scale equation}} & \forall w_b\in\mathcal{W}_b^0 \end{cases} \tag{3.30}$$

Firstly, we wish to solve the fine-scale equation. Then, we may represent $(u_{\varepsilon b}, w_b)$ as a function of b_K and $(u_{\varepsilon h}, w_h)$. Then, substituting the results of the bubble part $(u_{\varepsilon b}, w_b)$ in the large scale problem allows the determination of $u_{\varepsilon h}$.

3.2.3.4 Numerical examples

In this section, we present an example of the redistancing approach that has been implemented by solving the Equation (3.6) with a stabilized $SUPG$ finite element. The different steps of the followed algorithm are given below:

Input: Segmented image $\widehat{u_{seg}}$, initial mesh $\mathcal{H}$, thickness and fictitious time step $(\varepsilon, \Delta\tau)$
Output: Redistanced function u_ε^τ.

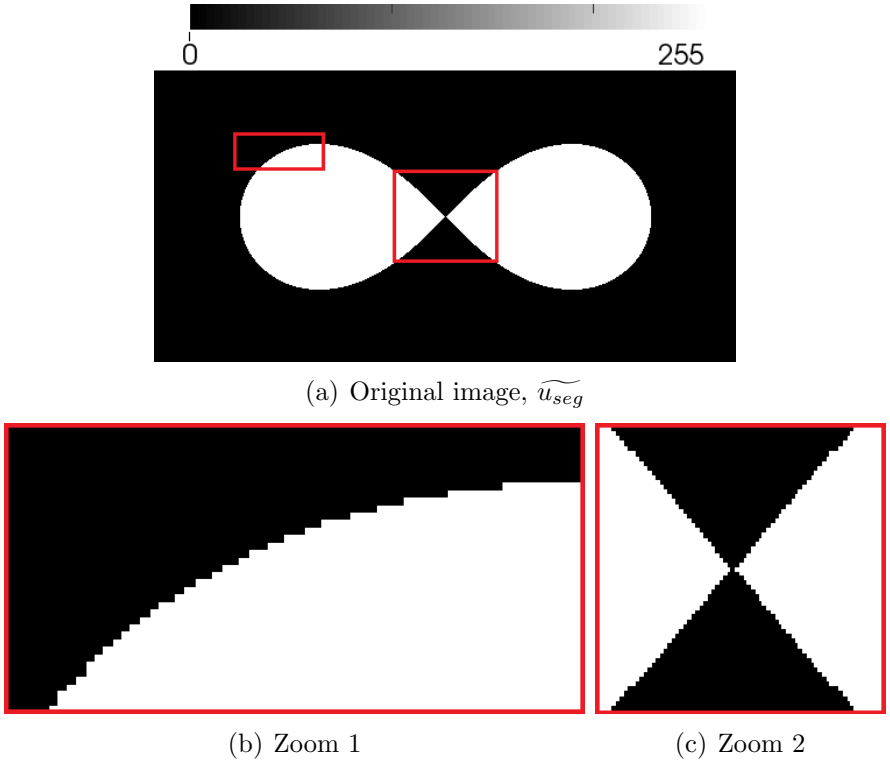
1 Interpolate the segmented image $\widehat{u_{seg}}$ on an initial mesh to obtain u_h. Then, let u_h be the initial distribution u_ε^0.
2 **while** *the redistanced function* u_ε^τ *has not converged* **do**
3 Compute the sign function S from the original image, $\widehat{u_{seg}}$, and interpolate it on the current mesh.
4 Solve the redistancing Equation (3.6), to obtain $u_\varepsilon^\tau(\varepsilon, S, \tau)$.
5 Correct the sign of $u_\varepsilon^\tau(\varepsilon, S, \tau)$, using the sign function S, as given by Equation (3.8).
6 $\tau = \tau + \Delta\tau$, update the redistanced function u_ε^τ.

First, we create a segmented image, $\widehat{u_{seg}}$, as shown in Figure 3.5(a) representing our example of size $(\widehat{L} \times \widehat{H} = 500 \times 250 = \widehat{T})$. This image has been obtained by drawing the values of the Lemniscate elliptic function:

$$\widehat{u}(Pixel^k) = \begin{cases} 255 & \text{if } \widehat{P^k} < 0 \\ 0 & \text{if } \widehat{P^k} \geq 0 \end{cases}$$

$$\text{and } \widehat{P^k} = ((\widehat{l^k}-250)^2 + (\widehat{h^k}-125)^2) + 2\cdot 125^2 \cdot ((\widehat{l^k}-250)^2 - (\widehat{h^k}-125)^2) \tag{3.31}$$

Zooming in the created image allows us to observe the hatched boundary, function of the pixel's size.



(a) Original image, $\widetilde{u_{seg}}$

(b) Zoom 1

(c) Zoom 2

Figure 3.5: Drawn image of the Lemniscate elliptic function and two progressive zooms, showing the pixel's size and its influence in the observations.

We interpolate this image onto an initial uniform mesh of size ($[0,1]\times[0,0.5]$, $N = 70000 < T$), by considering

$$u(\mathbf{X}^i) = \frac{\widehat{u}(Pixel^k)}{255} - 0.5, \tag{3.32}$$

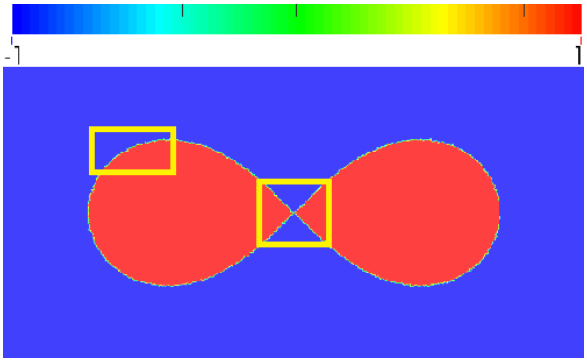
where $\widehat{l^k} = \text{int}(\frac{x^i}{\text{X}} \cdot (\widehat{L} - 1) + 1), \widehat{h^k} = \text{int}(\frac{y^i}{\text{Y}} \cdot (\widehat{H} - 1) + 1)$, $\forall i = 1, \ldots, N$, being the initial distribution defined as the initial interpolation, $u_h = u_\varepsilon^0(\varepsilon, S, \tau = 0)$. This initial value will not influence the final result, being the latter only dependent on the sign function S. Figure 3.6 shows the interpolation of the sign function on the initial mesh, S_h, using the same Lagrange interpolation operator from $\mathcal{V}$ to $\mathcal{V}_h$ as before:

Considering an uniform mesh and a P1 finite element approximation space, we plot the zero-level of the interpolation of the sign function S_h, as well as the boundary of the Lemniscate ellipse, computed from the segmented image $\widehat{u_{seg}}$ in Figure 3.7.

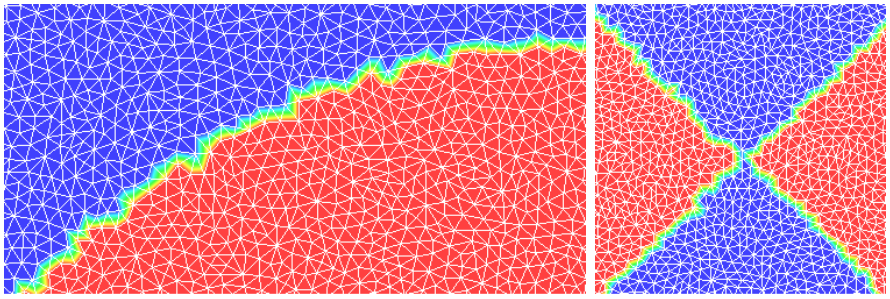
The zero-level of the interpolation of the sign function is obviously not the same as the object's boundary and shows its discontinuous character, following also the pixel's discretization. Let us perform the redistancing procedure, supposing different given thicknesses $\varepsilon = 0.01, 0.02, 0.03$ and without mesh adaptation. The fictitious time step $\Delta\tau$ necessary for the procedure depends on the thickness and follows:

$$\Delta\tau \approx c_\tau \cdot \varepsilon \tag{3.33}$$

where $c_\tau \in [0,1]$, giving $\Delta\tau = 0.00025, 0.0005, 0.00075$ for the chosen thickness.

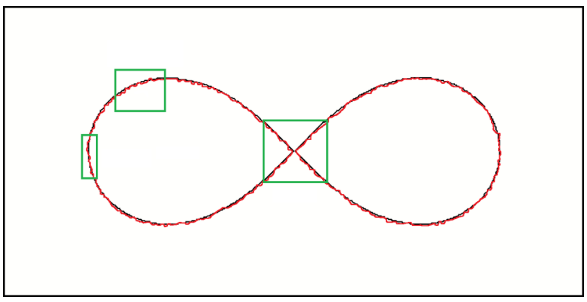


(a) S_h

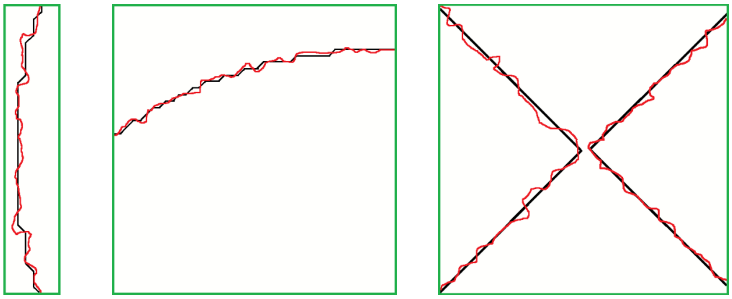


(b) Zoom 1 (c) Zoom 2

Figure 3.6: Interpolation of the sign function, S_h, based on an initial mesh with $N = 70000$ and two progressive zooms, showing certain discontinuities.



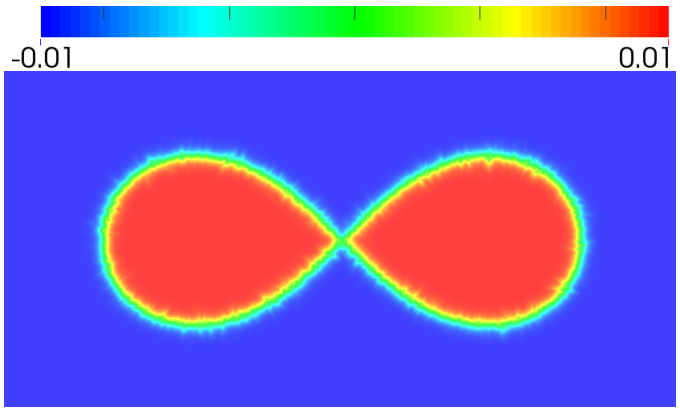
(a) Zero-level of S_h and boundary



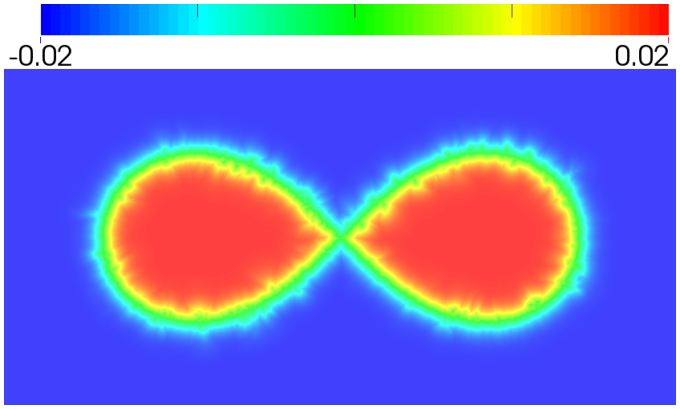
(b) Zoom 1 (c) Zoom 2 (d) Zoom 3

Figure 3.7: Comparison between the zero-level of S_h (red line) and the corresponding level of the Lemniscate elliptic function (black line).

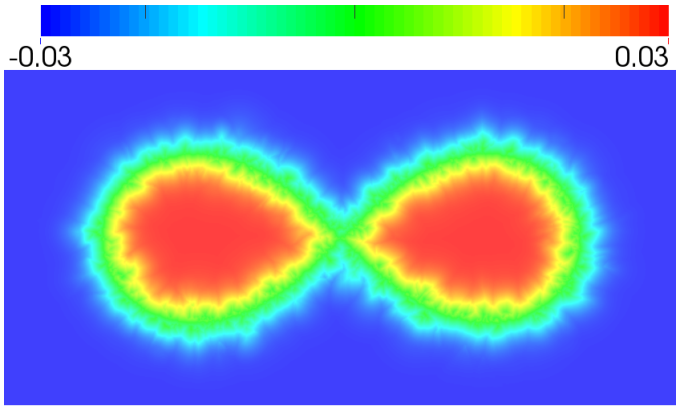
After $I \approx 60$ iterations, the redistanced function $u_\varepsilon^\tau(\varepsilon, S, \tau)$ reaches convergence, no longer evolving. The final results for the different thicknesses are illustrated in Figure 3.8.



(a) $u_\varepsilon^\tau, \varepsilon = 0.01$



(b) $u_\varepsilon^\tau, \varepsilon = 0.02$



(c) $u_\varepsilon^\tau, \varepsilon = 0.03$

Figure 3.8: Redistanced function solution when convergence is reached, u_ε^τ, for $\varepsilon = 0.01, 0.02, 0.03$.

As expected, the redistanced function distribution shows oscillations, both related with the mesh size and with the sign interpolated. To overcome with this problem, we choose to reduce the mesh size, in particular to improve the zero-level

of S_h. Automatic anisotropic mesh adaptation will be used for this purpose.

3.2.4 Redistancing method coupled to automatic anisotropic mesh adaptation

In this section, the redistancing procedure is coupled to anisotropic mesh adaptation. It will ensure a better representation of the zero-level of S_h, since the shape of the anisotropic elements may help capturing the gradient variation at lower computational cost, and the algorithm steps have been extended as follows:

Input: Segmented image $\widehat{u_{seg}}$, initial mesh $\mathcal{H}$, given thickness and fictitious time step $(\varepsilon, \Delta\tau)$, wanted number of nodes

Output: Redistanced function u_ε^τ and adapted anisotropic mesh

1 Interpolate the segmented image $\widehat{u_{seg}}$ onto an initial mesh to obtain initial values, $u_\varepsilon^0 = u_h$.

2 **while** *the redistanced function* u_ε^τ *has not converged* **do**

3 Compute the sign function S of the original image $\widehat{u_{seg}}$ on the current mesh.

4 Solve the redistancing equation, Equation (3.6) to obtain $u_\varepsilon^\tau(\varepsilon, S, \tau)$.

5 Correct the sign of $u_\varepsilon^\tau(\varepsilon, S, \tau)$ with the sign function S.

6 Generate the optimal mesh $\widetilde{\mathcal{H}}$ adapted to the redistanced function, $u_\varepsilon^\tau(\varepsilon, S, \tau)$.

7 $\tau = \tau + \Delta\tau$, update u_ε^τ and the mesh $\widetilde{\mathcal{H}}$.

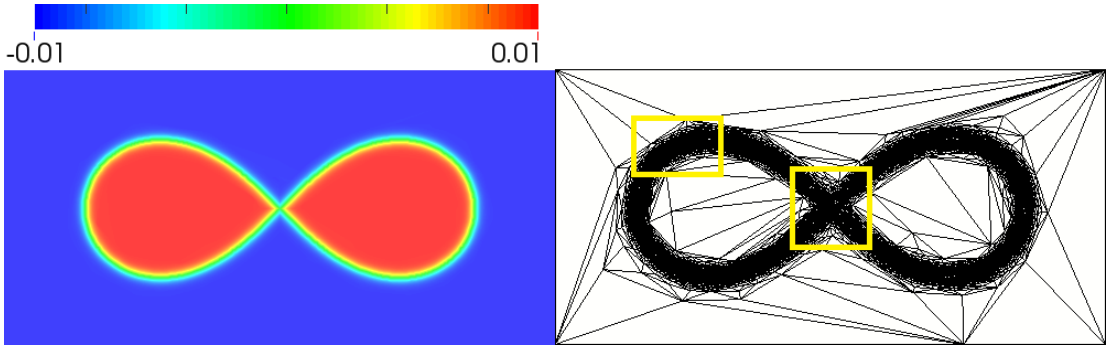
---

u_ε^τ is thus built simultaneously with its adequate mesh, by performing iterations with both features until convergence. The mesh number of nodes is fixed, to $N =$ 10000, but different values of $\varepsilon =$(0.01, 0.02, 0.03), and thus $\Delta\tau =$(0.00025, 0.0005, 0.00075) are considered. Figures 3.9(a), (b) and (c) illustrate the good enrichment of the mesh around the object's boundary, after $I = 60$ iterations.

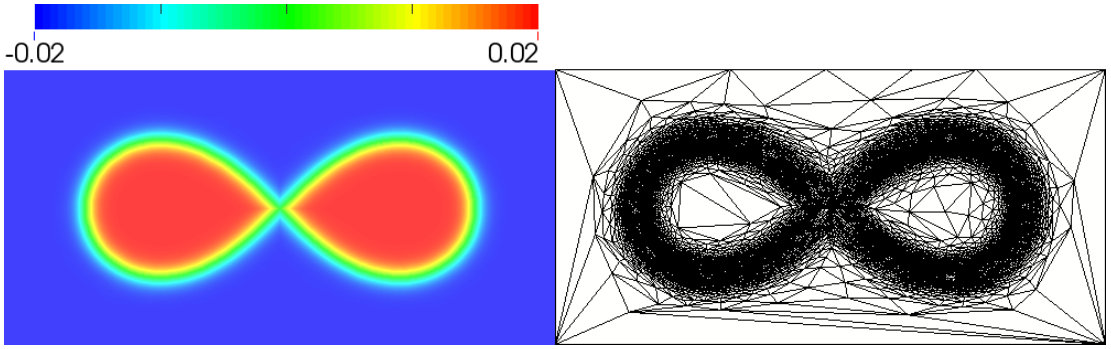
Figure 3.10 presents a plot of the zero-level of the sign function S_h (in red), with a zoom of the anisotropic mesh of the case $\varepsilon = 0.01$ shown in Figure 3.10.

One observes that oscillations are still reduced to the element size, but are much lower because the mesh size in the gradient direction is also smaller. The segmented image is drawn using Equation (3.31) and the signed distance function, $\widehat{u_d}$, can be easily computed and is illustrated in Figure 3.11(a). Image size is (500×250). This distance function is interpolated in a mesh of size $[0, 1] \times [0, 0.5]$, using also the relation $u_d = \frac{\widehat{u_d}}{500}$, as presented in Figure 3.11(b). The analytical hyperbolic tangent function, $u_\varepsilon^{anal}(u_d, \varepsilon)$, for three different thicknesses $u_\varepsilon^{anal}(u_d, 0.01)$, $u_\varepsilon^{anal}(u_d, 0.02)$, $u_\varepsilon^{anal}(u_d, 0.03)$ is given in Figure 3.11(c)(d)(e). This analytically computed values will be used for comparison with the redistanced ones.

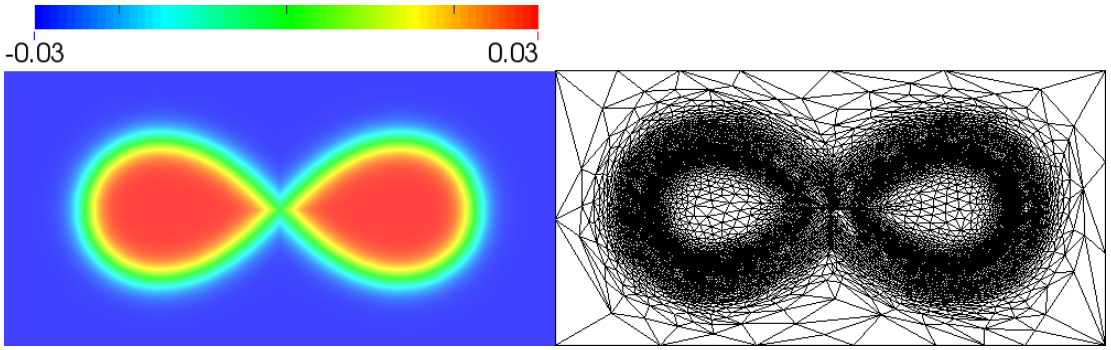
To study the convergence of the redistancing procedure, let us plot the MSE between the analytical hyperbolic tangent function, $u_\varepsilon(u_d, \varepsilon)$, and the redistanced one, $u_\varepsilon^\tau(\varepsilon, S, \tau)$, as a function of the number of performed iterations, for different thicknesses (Figure 3.12).



(a) $\varepsilon = 0.01$



(b) $\varepsilon = 0.02$

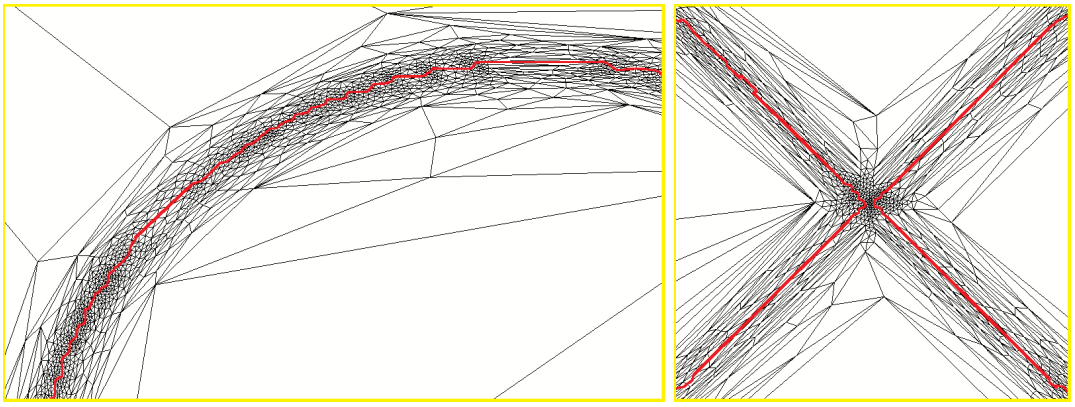


(c) $\varepsilon = 0.03$

Figure 3.9: Illustration of redistancing coupled to anisotropic adaptation, by presenting the final solution u_ε^τ and the corresponding mesh, for different values of ε: (a) $\varepsilon = 0.01$, (b) $\varepsilon = 0.02$, (c) $\varepsilon = 0.03$.

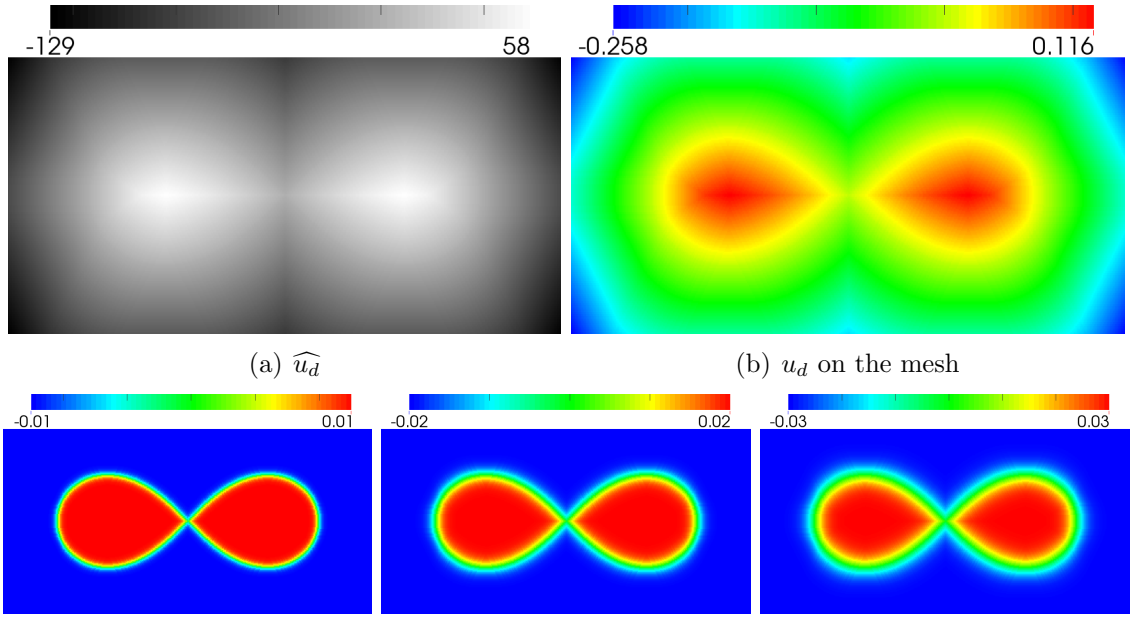
Curves with mesh adaptation have been drawn, showing that convergence is sooner attained after almost the same number of iterations. Smaller thicknesses generate also smaller errors. This is related to the fact that the adaptation strategy implies generating a fixed number of nodes on the thickness, ε, no matter what ε is, if the number of constrained nodes is high enough. Thus, the smaller ε is, the smaller mesh size and the smaller the error is. The number of iterations I to attain convergence can be approximated as

$$I \approx \frac{1.5\varepsilon}{\Delta\tau} \tag{3.34}$$



(a) Zoom 1 (b) Zoom 2

Figure 3.10: Zoom on the obtained anisotropic mesh and on the zero-level of S_h, for the case $\varepsilon = 0.01$.



(a) $\widehat{u_d}$ (b) u_d on the mesh

(c) $u_\varepsilon(u_d, 0.01)$ (d) $u_\varepsilon(u_d, 0.02)$ (e) $u_\varepsilon(u_d, 0.03)$

Figure 3.11: Image of the signed distance to the boundary of the segmented object, computed from $\widehat{u_{seg}}$, its interpolation on the mesh and the analytical hyperbolic tangent, with $\varepsilon = 0.01, 0.02, 0.03$, respectively in (c), (d) and (e).

The redistancing procedure starts from the zero-level, Γ, and progressively advances on both the positive and negative senses, until the contours reach 1.5ε and the gradient is well computed everywhere.

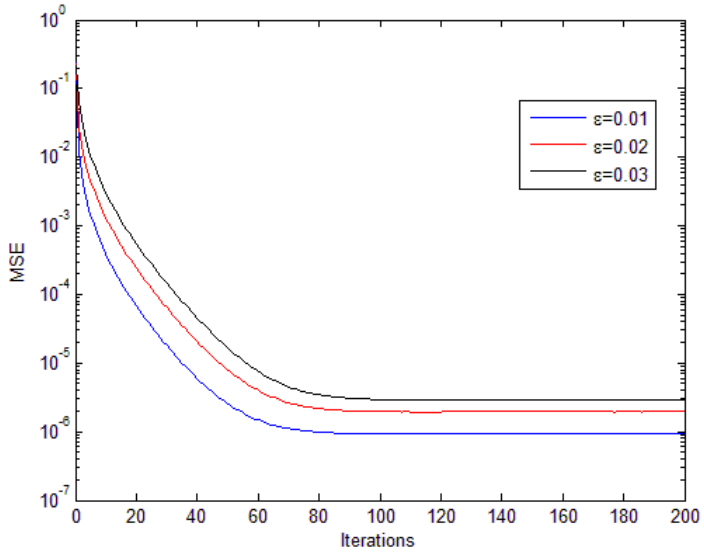


Figure 3.12: Illustration of the MSE as a function of the performed iterations, coupled with anisotropic adaptation, for different values of $\varepsilon = 0.01, 0.02, 0.03$

3.3 Image processing using mathematical morphology

The input image data is an important key to construct the mesh, whatever the image type is, either grey-scale, segmented or color. Image processing is very important. In this thesis, we have used an image processing tool, "Morph-M" [64]. Morph-M is a scientific library of image processing algorithms [65], developed by researchers at the CMM (Center for Mathematical Morphology [66], MINES ParisTech). Morph-M is implemented on C++ and contains most of the operators based on mathematical morphology to perform basic operations, such as dilation, erosion, opening and closing [67, 68], up to the most powerful operators, such as the hierarchical watershed [69]. Morph-M can be used through a Python interface, and be combined with other packages available through Python.

3.3.1 Introduction to Mathematical Morphology

Mathematical Morphology was first introduced in 1964 by Georges Matheron and Jean Serra at MINES ParisTech. The first research center for Mathematical Morphology [66] was founded in Fontainebleau.

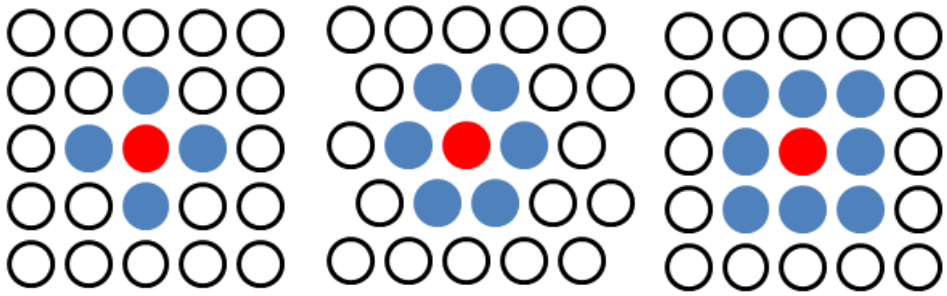
Mathematical Morphology is a mathematical theory and technique, informational structure analysis, which is related with algebra, lattice theory, topology and probabilities. The development of mathematical morphology is inspired by image processing problems, and is mainly applied to digital images. This technique consists of a set of operators that transform images according to the characterizations, such as size, shape, convexity, connectivity, or geodesic distance.

Hence, let us introduce of several image processing techniques that have been

used in this work, such as image segmentation, construction of the distance function or image gradient computation.

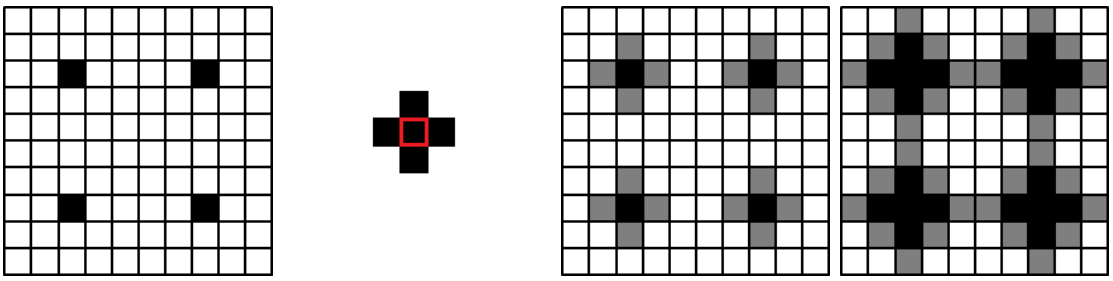
The basic operators of mathematical morphology are the erosion and the dilation, denoted as ϵ and δ. Both of them have firstly been developed to handle binary images and then extended to grey-scale ones.

Let us briefly introduce the idea behind the erosion and dilation algorithm. First, we introduce a "simplex" which is a pre-defined shape, used to fit or withdraw the objects in the image. This "simplex" is called the structuring element, noted as B. Usual structuring elements present 4-connexity, 6-connexity and 8-connexity (Figure 3.13). Let us consider A, a binary image in the Euclidean space $\mathbb{R}^d$, and consider the structuring element B. The following simple example shows the erosion algorithm on a binary image, centered at the origin O. A is of dimension (10×10) and is illustrated in Figure 3.14(a), with a set of 4-connexity elements B (Figure 3.14(b)). We go through all the pixels of A and check if they are coincident with the value of the center of B. If yes, the pixel is retained, otherwise it becomes black. The first and second erosion increments on the binary image are shown in Figures 3.14(c) and (d).



(a) 4-connexity (b) 6-connexity (c) 8-connexity

Figure 3.13: Usual structuring elements.



(a) Binary image (b) 4-connexity element (c) 1^{st} erosion step (d) 2^{nd} erosion step

Figure 3.14: Illustration of erosion of a binary image with a 4-connexity structuring element, as well as the obtained image in two erosion increments.[65]

On the other hand, the dilation algorithm uses a black set of element B to modify the binary image, with the same idea as the erosion algorithm. For a grey-scale image, let us represent images as space functions, $f(x)$. The erosion and dilation of

this function by a flat structuring element B is the dilation and erosion of each set $b(x)$ by B, as the plane. An example is given in Figure 3.15.

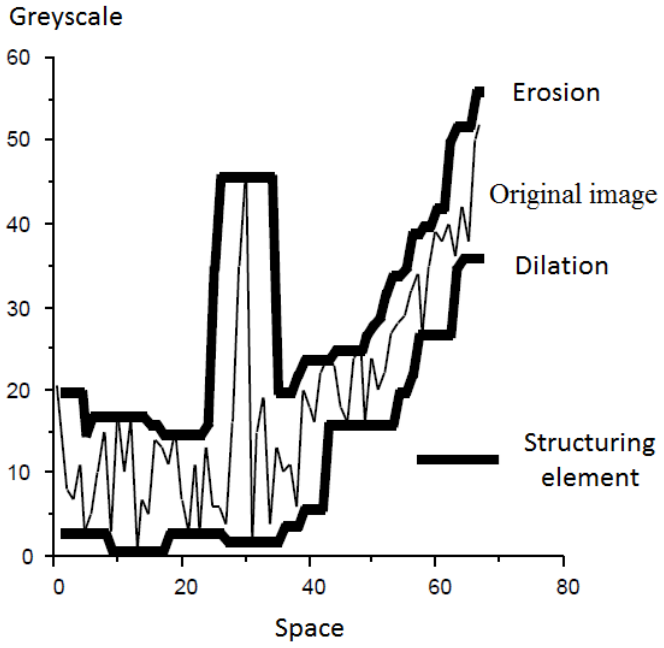
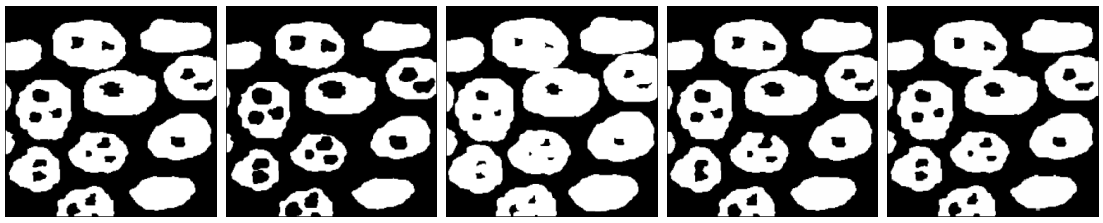


Figure 3.15: Illustration of erosion and dilation on a grey-scale image, represented by a function $f(x)$

In other words, erosion shrinks the positive peaks and those thinner than the structuring element disappear. It expands the valleys and the sinks, whereas dilation produces the dual effect.

Opening and closing are other basic algorithms for morphological noise segmentation. Opening segments small objects, whereas closing removes small holes. The opening (γ) and closing (φ) couple to erosion and dilation by performing $\gamma = \delta\epsilon$ and $\varphi = \epsilon\delta$. Furthermore, opening and closing are idempotent: $\gamma\gamma = \gamma$ and $\varphi\varphi = \varphi$.

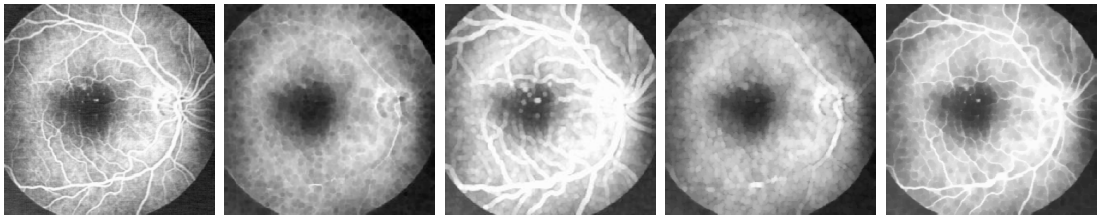
As an example, processing of a binary and grey-scale image by erosion, dilation, opening and closing is illustrated in Figures 3.16 and 3.17.



(a) Original binary image A (b) Erosion $\epsilon(A)$ (c) Dilation $\delta(A)$ (d) Opeing $\gamma(A)$ (e) Closing $\varphi(A)$

Figure 3.16: Illustration of erosion, dilation, opening and closing of binary image A

These morphological techniques are thus efficient ways to process images.



(a) Original grey-scale image A (b) Erosion $\epsilon(A)$ (c) Dilation $\delta(A)$ (d) Opening $\gamma(A)$ (e) Closing $\varphi(A)$

Figure 3.17: Illustration of erosion, dilation, opening and closing of a grey-scale image A.

3.3.2 Regularized function construction

In the image redistancing procedure, we build a regularized level-set function, which requires many iterations to attain convergence, in particular for 3D images. Instead, one can directly compute a signed distance function, using mathematical morphology operators. Such a distance, interpolated in the mesh, will still show the trace of the pixelation/voxelation. Then, redistancing will be necessary but starting from this morphologically modified image.

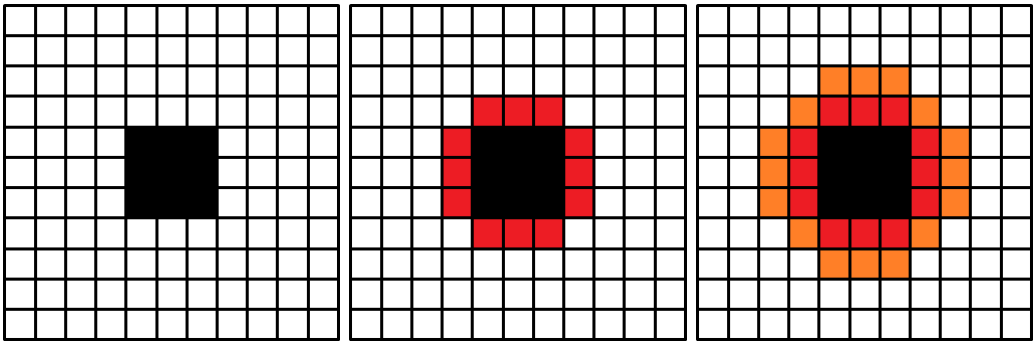
To illustrate this point, let us build the signed distance function of a binary image $\widehat{u}$ of dimensions (11×11)(Figure 3.18(a)), using the previously presented 4-connexity element and an erosion/dilation algorithm. For example, the first erosion step, from the original image, will find the most approximate pixels, like shown in Figure 3.18(b). The second erosion step will find the adjacent layer, based on the previous results, as given in Figure 3.18(c). In this case, the $7th$ erosion step will cover the entire image (Figure 3.18(d)). On the other hand, the first dilation step will find the complementary pixel region and, after two increments, the result is the one shown in Figure 3.18(e). Finally, by combining the two results (both of erosion and dilation), we may consider that the distance of the object coming from the segmented image, $\widehat{u}_d$, is built.

3.3.3 Image gradient computation

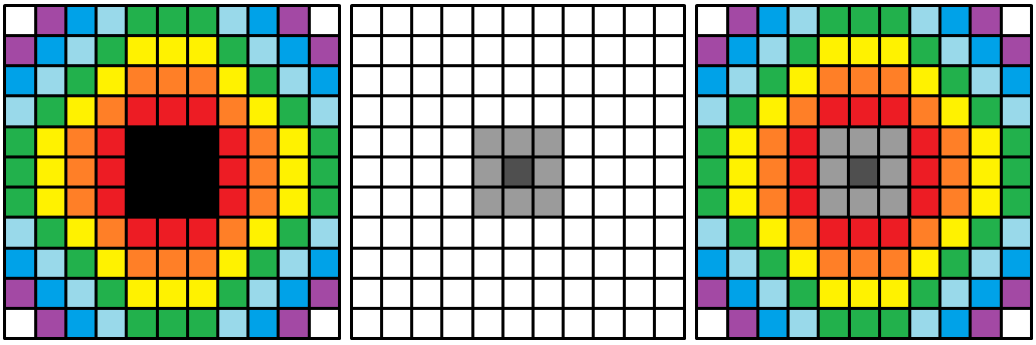
Another application concerns the determination of morphological gradients. The goal of using gradient transformations is to be able to highlight image contours. Based on the erosion and dilation algorithms and by comparing the modified and the original images, the residual values can be seen as contours, very easy to handle. Figure 3.19 illustrates the basic idea and the interest behind the methodology.

This method is useful to obtain a description of the boundary of an object of the image.

Later in this chapter, we will see that using these segmentation tools will allow building the closest initial solution as possible for our redistancing procedure.



(a) Original image A (b) 1^{st} erosion (c) 2^{nd} erosion



(d) 7^{th} erosion (e) 2^{nd} dilation (f) Distance $\widehat{u_d}$

Figure 3.18: Computation of the signed distance to an object in an original image.

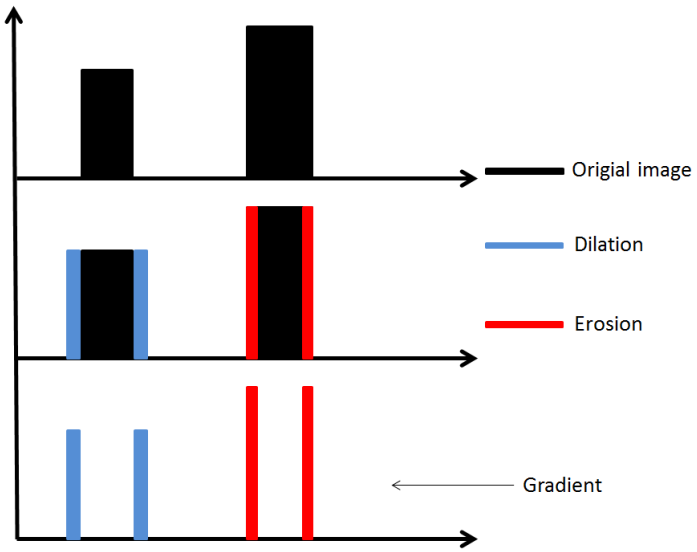


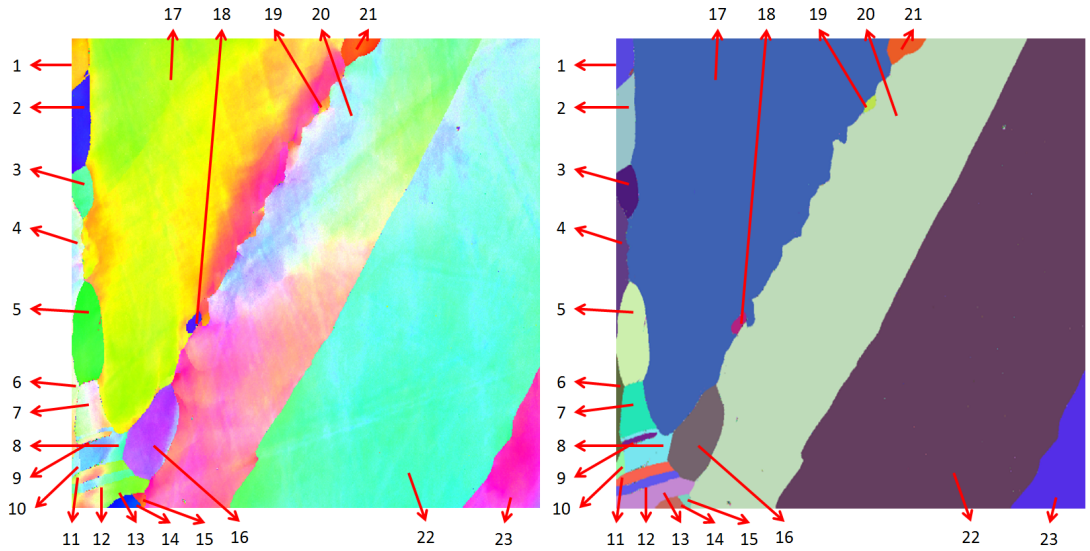
Figure 3.19: Illustration of erosion and dilation use to detect image gradients.

3.4 Numerical examples

3.4.1 2D color images

To distinguish the phases, image segmentation is often a compulsory step. In the previous chapter, we have presented adaptive anisotropic adaptation applied directly

to an RGB polycristal image, Figure 3.20(a), which does not accurately represent the interfaces between the different crystals. Let us consider this example, based on the three gradients of RGB channels. A segmented polycristal image containing 23 different colors, representing 23 different crystalline orientations need to be traced (as seen in Figure 3.20(b)).



(a) Original polycristal image (b) Segmented polycristal image

Figure 3.20: Polycristal segmentation from the original polycristal image, using 23 colors.

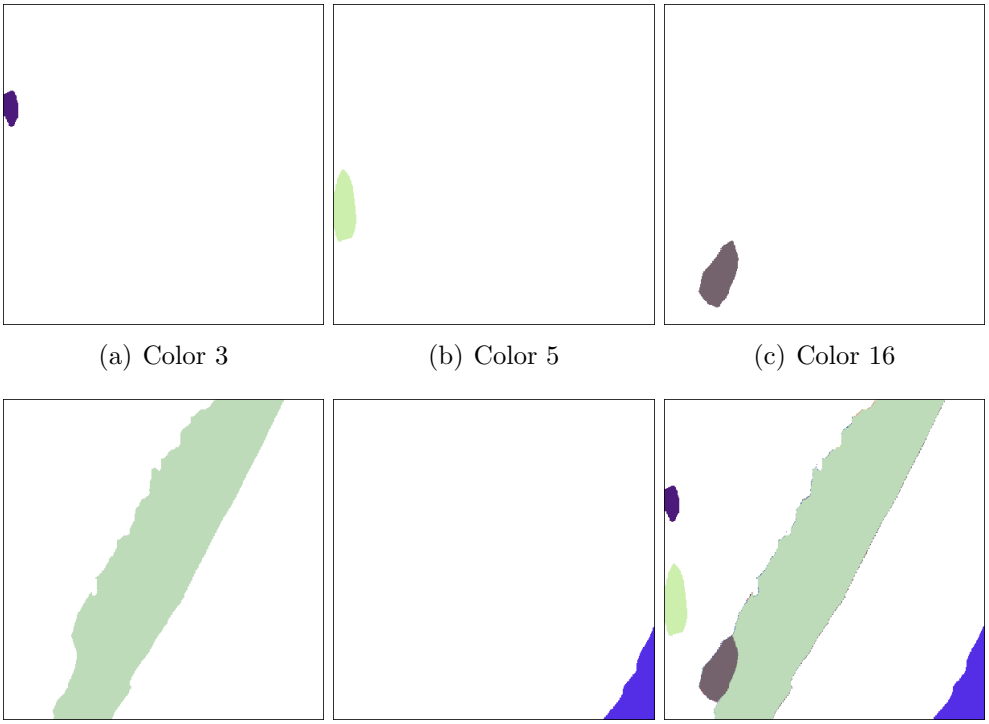
If we decompose the color image into the three R, G and B channels, it will not satisfy our query (identify adequately the phases and have an optimal mesh). Therefore, we propose a new method, designated "multi-pixel/voxel-channels", distinguishing the object through its specific pixel/voxel's values. Each $Pixel$ has three channels, $\mathbf{Pixel} = \{Pixel_{red}, Pixel_{green}, Pixel_{blue}\}$. We consider the three values of the three channels to define the color we wish to map and if the images pixel correspond to its value, we fix it as this color. For the example proposed in Figure 3.20, let us suppose five **Pixels**, with the three channels' values presented in Table 3.1.

Selected **Pixels**	*Red* channel	*Green* channel	*Blue* channel	Selected color
$\mathbf{Pixel}_1$	76	26	123	Color 3
$\mathbf{Pixel}_2$	204	239	173	Color 5
$\mathbf{Pixel}_3$	116	99	109	Color 16
$\mathbf{Pixel}_4$	190	219	185	Color 20
$\mathbf{Pixel}_5$	84	46	231	Color 23

Table 3.1: Selected colors and the three channels' values.

Selected colors are shown in Figure 3.21(a) to (e), the total selected objects on Figure 3.21(f).

These five colors are interpolated on an initial mesh, where they are represented by five functions gathered in the vector $\overrightarrow{u_h} = \{u_{h(1)}, u_{h(2)}, u_{h(3)}, u_{h(4)}, u_{h(5)}\}$. Then,



(a) Color 3 (b) Color 5 (c) Color 16

(d) Color 20 (e) Color 23 (f) Total selected colors

Figure 3.21: Five selected colors.

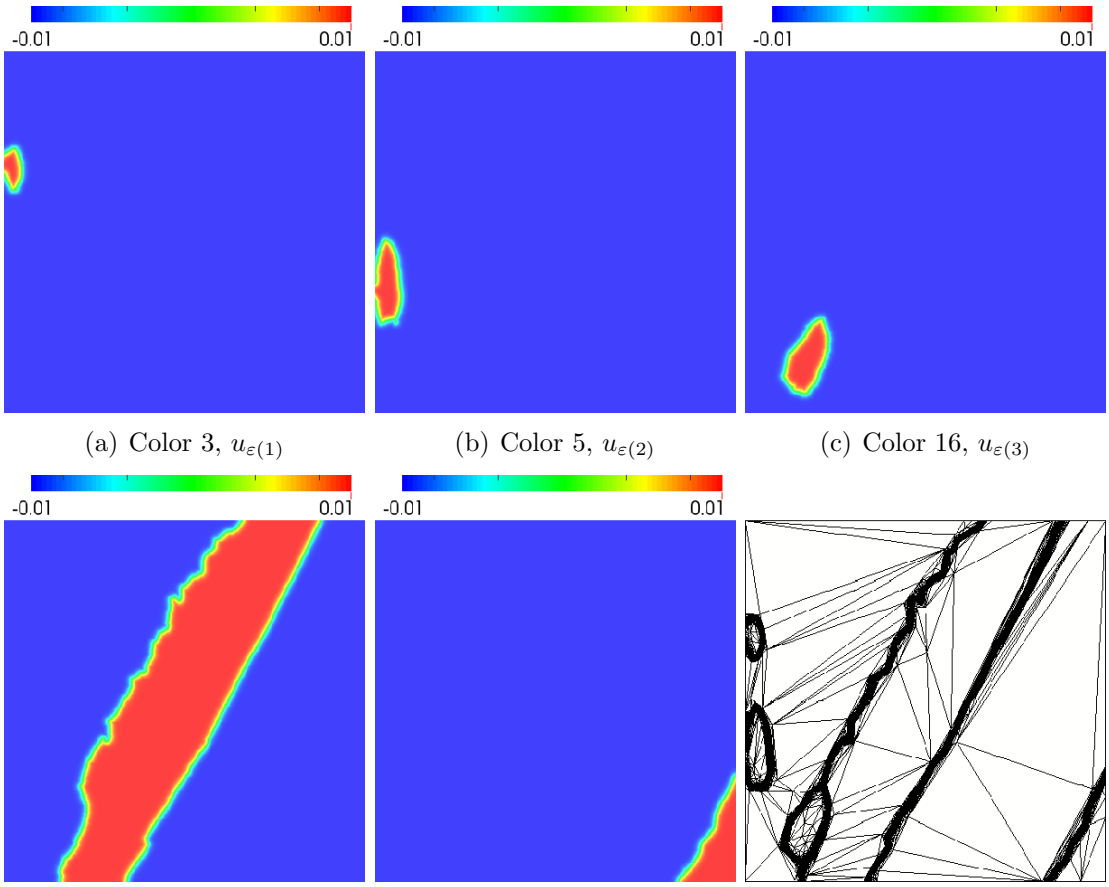
the redistancing-adaptation procedure is applied in a multi-component context, to $\overrightarrow{u_\varepsilon^\tau} = \{u^\tau_{\varepsilon(1)}, u^\tau_{\varepsilon(2)}, u^\tau_{\varepsilon(3)}, u^\tau_{\varepsilon(4)}, u^\tau_{\varepsilon(5)}\}$. The algorithm steps followed are given below:

Input: Segmented color image data, $\widehat{u_{seg}}$, selected colors, through the **Pixels**' vector, initial mesh and the given number of nodes N for mesh adaptation, (ε, τ) for the redistancing procedure.

Output: Redistanced function $u^\tau_{\varepsilon(i)}$ and adapted anisotropic mesh

for *each* $\boldsymbol{Pixel}_{(i)}$ **do**
  Distinguish the corresponding $\mathbf{Pixel}_{(i)}$'s value, interpolate the segmented image $\widehat{u_{seg}}$ onto an initial mesh to obtain the initial function $u^0_{\varepsilon(i)} = u_{h(i)}$.
while *the redistanced function* $u^\tau_{\varepsilon(i)}$ *does not converge* **do**
  for *each* $\boldsymbol{Pixel}_{(i)}$ **do**
    Compute the sign function $S_{(i)}$ from the original image $\widehat{u_{seg}}$ on the current mesh.
    Solve the redistancing equation (3.6), to obtain $u^\tau_{\varepsilon(i)}(\varepsilon, S_{(i)}, \tau)$.
    Correct the sign of $u^\tau_{\varepsilon(i)}(\varepsilon, S_{(i)}, \tau)$ using the sign function $S_{(i)}$.
  Generate the optimal mesh, adapted to the components of $\overrightarrow{u_\varepsilon^\tau}$.
  $\tau = \tau + \Delta\tau$, update the redistanced function $u^\tau_{\varepsilon(i)}$ and the new adapted mesh.

Let us input a thickness $\varepsilon = 0.01$, a fictitious time step $\Delta\tau = 0.00025$ and



(a) Color 3, $u_{\varepsilon(1)}$ (b) Color 5, $u_{\varepsilon(2)}$ (c) Color 16, $u_{\varepsilon(3)}$

(d) Color 20, $u_{\varepsilon(4)}$ (e) Color 23, $u_{\varepsilon(5)}$ (f) Adapted anisotropic mesh

Figure 3.22: Illustration of the five redistanced functions $u^{\tau}_{\varepsilon(i)}$ and the final adapted anisotropic mesh with $N = 40000$.

a number of nodes $N = 40000$. After about 60 iterations, the multi-redistanced function $\overrightarrow{u^{\tau}_{\varepsilon}}$ and the final adapted anisotropic mesh are shown in Figure 3.22.

Each redistanced function is independent from the others, which is the wanted situation for physical parameters definition and the adapted anisotropic mesh is well adapted the boundary of the chosen objects. In conclusion, using the proposed multi-pixel-channels framework coupled to the redistancing-adaptation method to handle color image performs better than the previous simple RGB-channel method, which could not distinguish the objects. Also, our methodology provides multi-redistanced functions on one adaptive anisotropic mesh, which may be used for numerical simulations, keeping an accurate representation.

3.4.2 2D Head MRI-Brain

To continue illustrating the usage of redistancing-adaptation, let us use the 2D head MRI image, represented again in Figure 3.23(a). As an example, we consider that we wish to segment the white matter of the brain region, identified in a non accurate way by our naked eyes. Image processing is illustrated in Figure 3.23, by applying closing, gradient and filtering operations, to obtain a segmented white matter in $\widehat{u_{seg}}$.

After this image processing procedure, we have applied the redistancing-adaptation operator. First, $\widehat{u_{seg}}$ has been interpolated onto an initial mesh of dimensions $[0,1] \times [0,1]$, by doing: $u_\varepsilon^0 = \widehat{u_{seg}}/255 - 0.5$. Since the segmented white matter is extremely complex geometrically, only small thicknesses may capture the boundary of this region. Therefore, we have input the thickness $\varepsilon = 0.005$ and the fictive time step $\Delta\tau = 0.00005$, for a number of nodes $N = 20000$, to be able to accurately describe the complexity of the boundary of the white matter without losing important detail. The sign function S of this region is shown in Figure 3.24(a). After approximately 150 iterations and 680 seconds of CPU time, the final results are illustrated in Figures 3.24, by plotting the redistanced function u_ε^τ and the adapted anisotropic mesh.

To quantitatively estimate the results, we plot a superposition of the object's boundary computed from the image (the 127.5-level of $\widehat{u_{seg}}$, blue line) and the zero-level of u_ε^τ (red line) in Figure 3.24(d). One observes a very good agreement, showing that the grey matter's definition has been perfectly preserved.

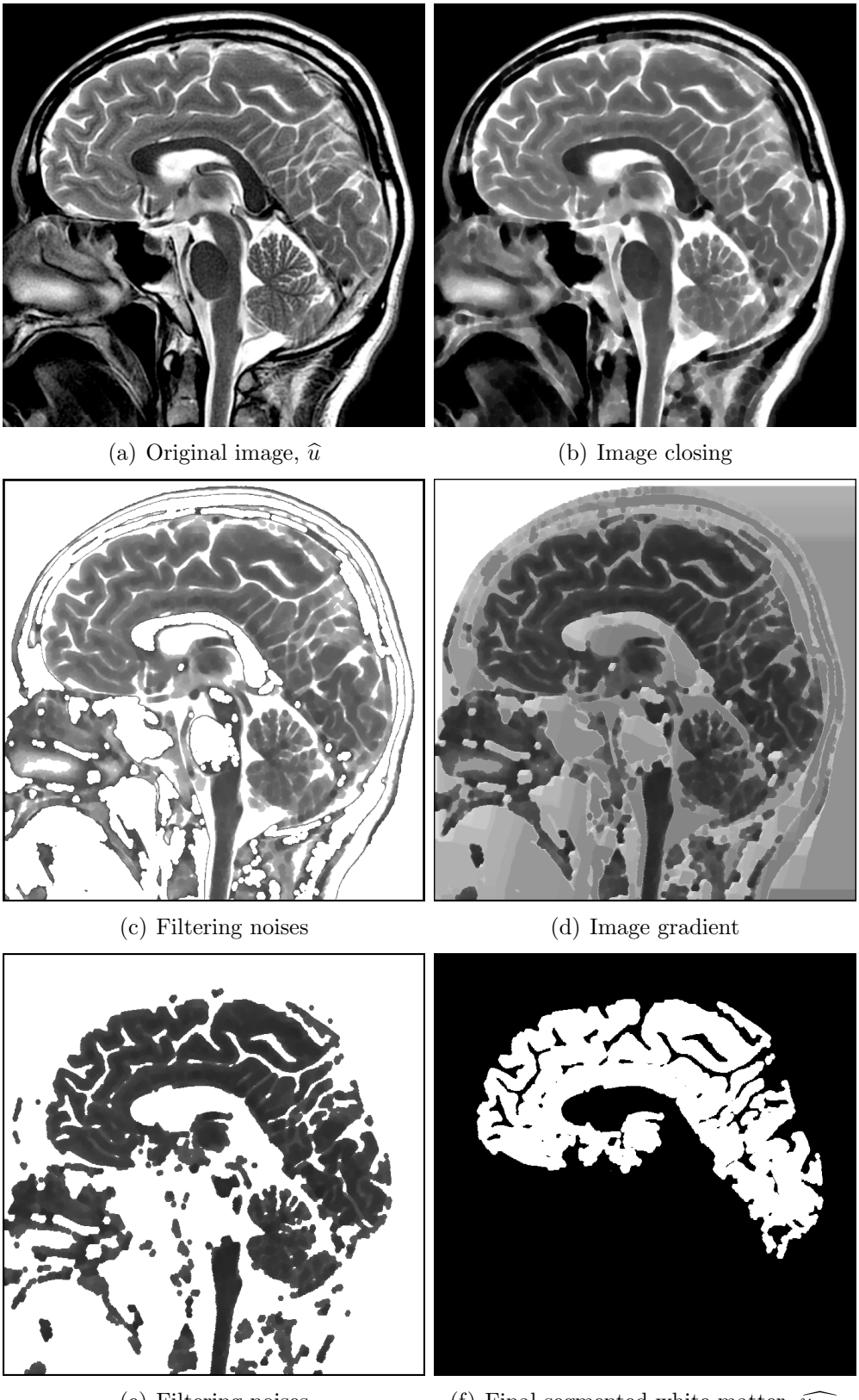
3.4.3 Sensitivity to the initial solution

The redistancing-adaptation method may be also extended to 3D applications using the same algorithm as in the 2D case. However, to converge to the redistanced function, hundreds of redistancing iterations coupled to hundreds of mesh adaptations may be required, which will cost extremely in terms of computing resources and CPU time. Therefore, we have decided to try to accelerate the redistancing procedure and reduce computing costs by attaining quickly the convergence using an optimal initial solutiom.

When solving the redistancing equation, an initial value $u_\varepsilon^0(S, \varepsilon, \tau = 0)$ does not influence the final result, since only the sign function S is important, but it has an impact on the number of performed iterations of the method and on the speed of convergence.

To optimize this point, Morph-M may be used to provide the initial distance or phase function as an image, based on Mathematical Morphology techniques. To demonstrate our point, let us first compute the signed distance of the 2D head MRI image (Figure 3.23(f)) in an image $\widehat{u_d}$ of dimensions (700×700), shown in Figure 3.25(a), computation that run in 4 seconds using Morph-M. Then, this image is interpolated on the initial mesh, of dimensions $[0,1] \times [0,1]$, to recompute the signed distance in the mesh, u_d:

$$u_d(\mathbf{X}^i) = \frac{\widehat{u_d}(Pixel^k)}{700} \tag{3.35}$$

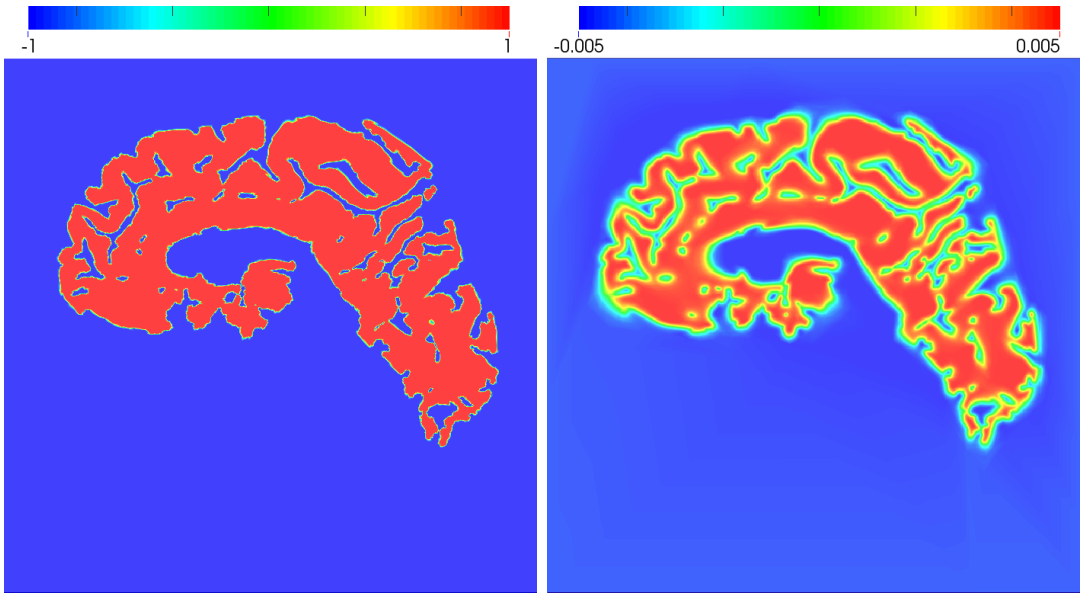


(a) Original image, $\widehat{u}$ (b) Image closing

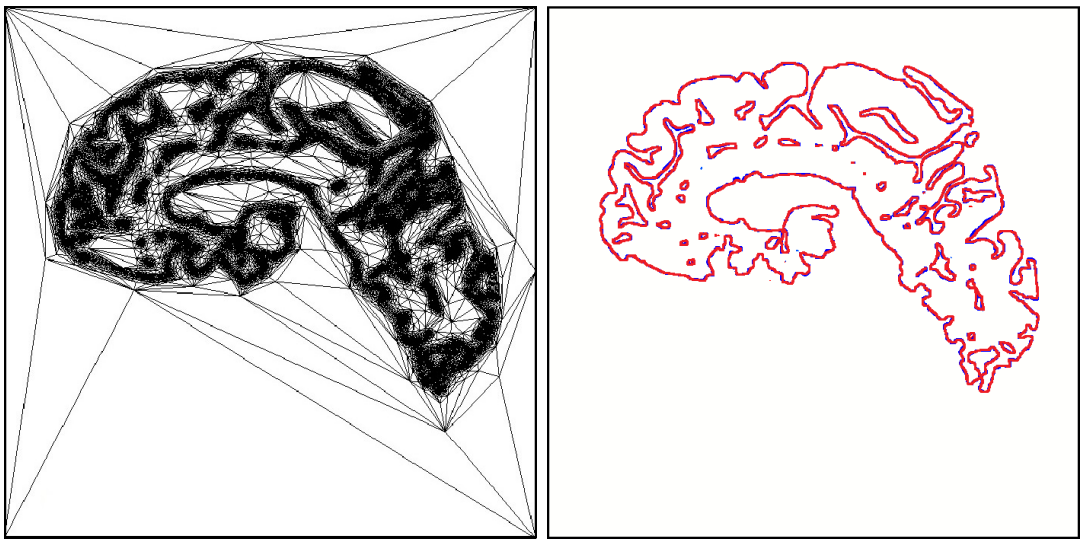
(c) Filtering noises (d) Image gradient

(e) Filtering noises (f) Final segmented white matter, $\widehat{u_{seg}}$

Figure 3.23: Image processing of the 2D head MRI image, to get the segmented white matter.



(a) Sign function, S (b) Redistanced function, u_ε^τ



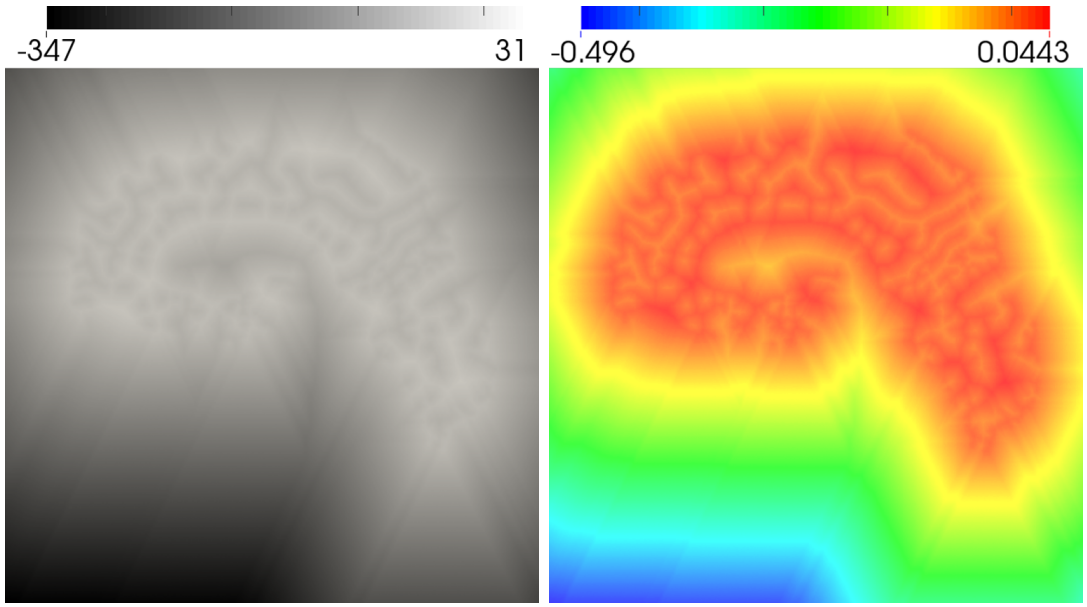
(c) Adapted anisotropic mesh, $N = 20000$ (d) Zero-level of u_ε and 127.5-level of $\widehat{u_{seg}}$

Figure 3.24: Illustration of the redistancing-adaptation procedure on a 2D segmented head MRI, building a smooth phase function corresponding to the brain's white matter.

where $\widehat{l^k} = \text{int}(\frac{x^i}{\text{X}} \cdot (\widehat{L} - 1) + 1), \widehat{h^k} = \text{int}(\frac{y^i}{\text{Y}} \cdot (\widehat{H} - 1) + 1)$, as seen in Figure 3.25(b).

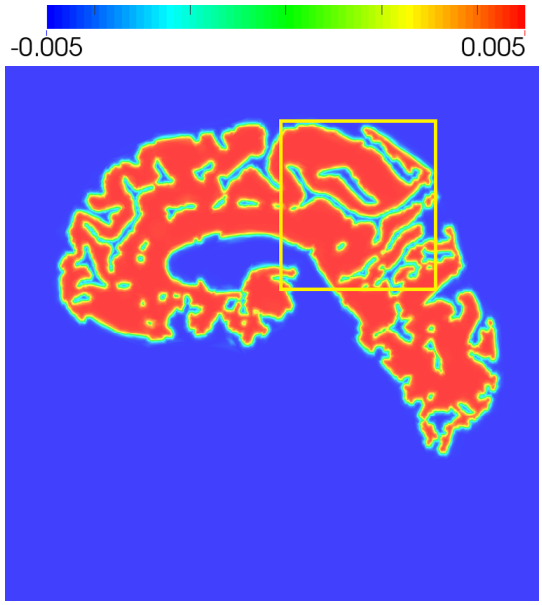
Knowing the signed distance function on the mesh, u_d (as seen in Figure 3.25(b)), its hyperbolic tangent function $u_\varepsilon(\varepsilon, u_d)$ is easily given by considering, for example, $\varepsilon = 0.005$, as drawn in Figure 3.25(c). This is the initial u_ε^0.

We have performed two different test cases. For the first one, only mesh adaptation was launched with $N = 20000$, with a convergence after 15 iterations of 88 seconds CPU time. Test cases were driven on one core (3.5GHz,16 Gb RAM). Figure

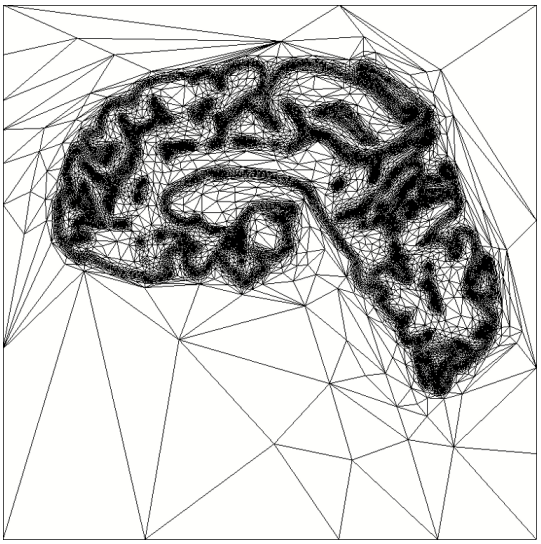


(a) $\widehat{u_d}$

(b) u_d on the mesh



(c) $u_\varepsilon^0 = u_\varepsilon(u_d, \varepsilon = 0.005)$



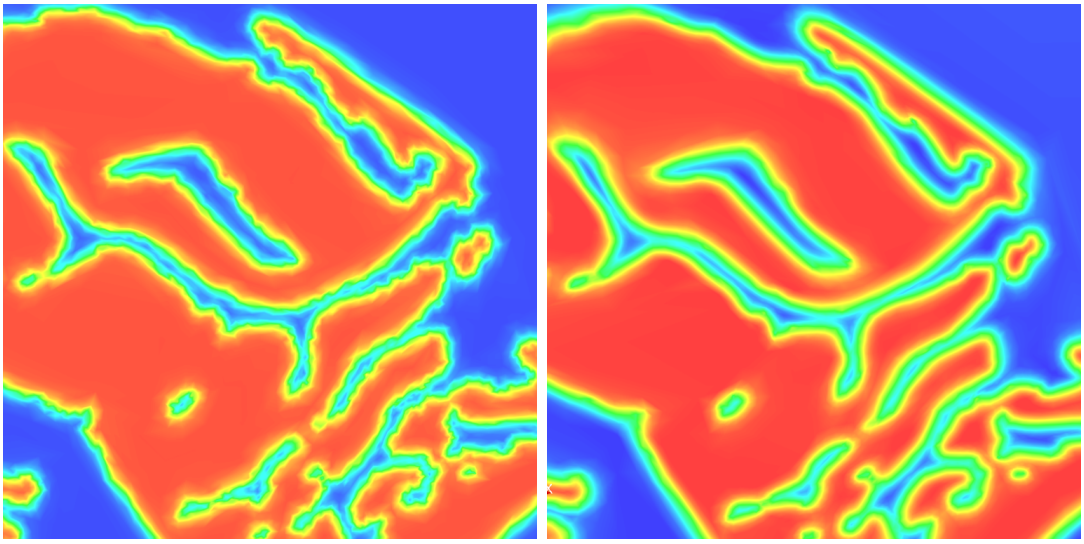
(d) Adapted $u_\varepsilon(u_d, \varepsilon = 0.005)$ anisotropic mesh, $N = 20000$

Figure 3.25: Illustration of the distance computation on the image, initial computed hyperbolic tangent function $u_\varepsilon^0 = u_\varepsilon(u_d, \varepsilon = 0.005)$, and adapted anisotropic mesh.

3.25(d) presents the obtained anisotropic mesh.

In the second, the redistancing-adaptation procedure was also performed for the same number of nodes and iterations, with a CPU time of 93 seconds also on one core, a little longer than when only mesh adaptation was performed, since the initial function is very close to the expected solution. Results are the ones that illustrated in Figure 3.24.

To compare the results of these two cases, a zoom was performed and drawn in Figure 3.26 where, on the left, we plotted the phase function computed from image, $u_\varepsilon(u_d, \varepsilon)$ (Figure 3.26(a)) and the redistancing-adaptation solution u_ε^τ (Figure 3.26(b)). We observe that the redistanced function, u_ε^τ, is more smooth than the non-redistanced one, where we still find a trace of the pixels, less adapted for the upcoming numerical simulations. Thus, for a very small increase on the computation cost, we have decided to perform: (1) distance computation using Morph-M, $\widehat{u_d}$;(2) interpolation of $\widehat{u_d}$ on the mesh to get u_d;(3) $u_\varepsilon^0(u_d, \varepsilon)$ computation;(4) redistancing-adaptation to obtain u_ε and anisotropic mesh.



(a) Zoom of $u_\varepsilon(u_d, \varepsilon)$ (b) Zoom of $u_\varepsilon^\tau(\varepsilon, S, \tau)$

Figure 3.26: Solution of u_ε obtained (a) by computing it directly from the image, $u_\varepsilon(\varepsilon, \widehat{u_d})$; (b) by redistancing it $u_\varepsilon(\varepsilon, S, \tau)$.

Additionally, the most important conclusion is that it may significantly reduce the redistancing time, especially in 3D cases.

3.4.4 3D head MRI

MRI scanning provides an inside image of the head, which is very important in medical research and therapy. From the information contained in these images, some may be used for different types of simulation. Skull can be the object of solid mechanics studies, craniotomy operations or impact tests. One other example concerns the study of the behavior of the Cerebrospinal fluid (CSF), where simulation is also very relevant. CSF is the corporal fluid in the head, between the skull and the brain, proving basic mechanical and immunological protection. Dynamic of the

CSF may be treated using classical computational fluid mechanics to obtain, for example, the pressure in the brain, ... Grey matter and white matter are the central nervous system of the human body, controlling all the human behavior, many researches on them are still at the primary stage. Therefore, head image processing is a very wide research field. On the other hand, simulation of on all of these topics based on images and meshes may be helpful for medical knowledge improvement.

In this section, the image processing and redistancing-adaptation method described before will show its potential in this field. Hence, a segmented 3D human head MRI, $\widehat{u_{seg}}$, Figure 3.27, is studied and is composed of the following structures: muscles/skin, skull, cerebrospinal fluid, grey matter and white matter. The 3D image has been segmented by [40, 41], on an image of size ($\widehat{L}\times\widehat{H}\times\widehat{W} = 255\times255\times180$). Representations of the voxels' value in the segmented image is detailed in Table 3.2.

Value of $\widehat{u_{seg}}(Voxel)$	Represented part of the head
0	Outer part
1	Skin, eyes, muscle
2	Skull
3	CSF
4	Grey matter
5	White matter

Table 3.2: Representation of the voxels' value in the 3D human head MRI image.

Figure 3.27 shows the segmented image with the six different voxels' values.

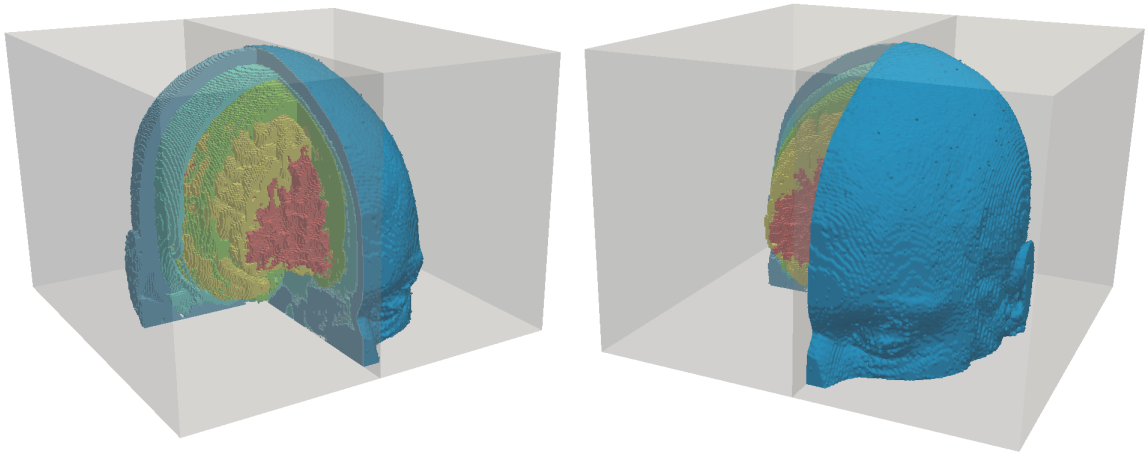


Figure 3.27: 3D segmented MRI of the human head.

We have decided to consider three regions: the whole head (1+2+3+4+5), the skull (2) and the brain (4+5). A thresholding was done to segment the image and obtain $\widehat{u_{head}}$ (the whole head), $\widehat{u_{skull}}$ (the skull), $\widehat{u_{brain}}$ (the brain).

$$\widehat{u_{head}}(Voxel)\begin{cases}255 & \text{if } \widehat{u_{seg}}(Voxel) > 0\\ 0 & \text{if } \widehat{u_{seg}}(Voxel) = 0\end{cases}$$

$$\widehat{u_{skull}}(Voxel)\begin{cases}255 & \text{if } \widehat{u_{seg}}(Voxel) = 2\\ 0 & \text{if } \widehat{u_{seg}}(Voxel) \neq 0\end{cases} \tag{3.36}$$

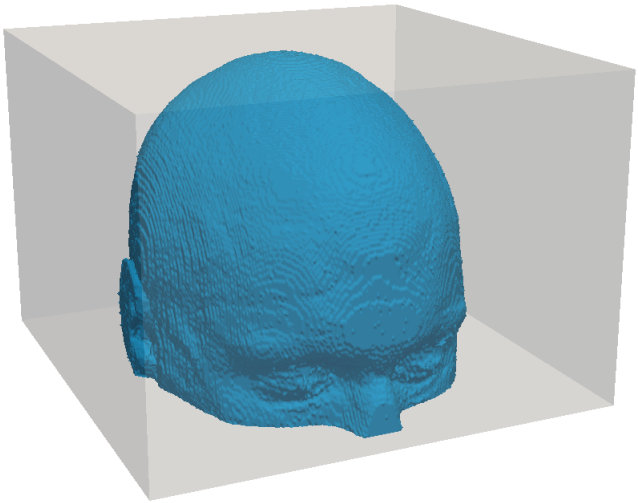
$$\widehat{u_{brain}}(Voxel)\begin{cases}255 & \text{if } \widehat{u_{seg}}(Voxel) > 3\\ 0 & \text{if } \widehat{u_{seg}}(Voxel) \leq 3\end{cases}$$

After the image processing step, where these three parts were easily segmented, they were drawn in Figure 3.28.

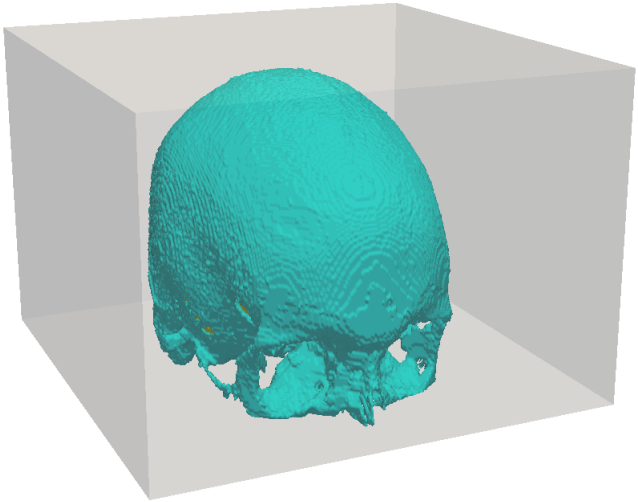
Then, as presented in the previous section ,using the signed distance to the boundary of the object segmented in these images, $\widehat{u_{d_{head}}}$, $\widehat{u_{d_{skull}}}$, $\widehat{u_{d_{brain}}}$ were obtained using Morph-M, with a CPU time of only 8 seconds for each case. The final result is shown in Figures 3.29(a)(b)(c). These three signed distance images were interpolated onto an initial mesh with dimensions $[2.55 \times 2.55 \times 1.79]$, by considering $u_d(\mathbf{X}^i) = \widehat{u_d}(Voxel^k)/100$, to obtain the distance functions on the mesh. For a thickness $\varepsilon = 0.02$, the initial function $u_\varepsilon^0(\varepsilon, S, \tau) = u_\varepsilon(u_d, \varepsilon)$ was computed and is plotted in Figure 3.29.

In fact, the more complex geometries ask for more nodes for a correct description, so the imposed number of nodes for the head, skull and brain cases are, respectively, $N = 200000, 300000, 400000$. After 25 redistancing-adaptation iterations, run on 6 cores, the obtained results are illustrated in Figures 3.30 and 3.31, representing the redistanced function u_ε^τ, the adapted anisotropic volume mesh, the iso-zero surface of u_ε and the surface mesh on this iso-zero surface. The CPU time for each case was: 148.4, 183.6 and 233.9 minutes.

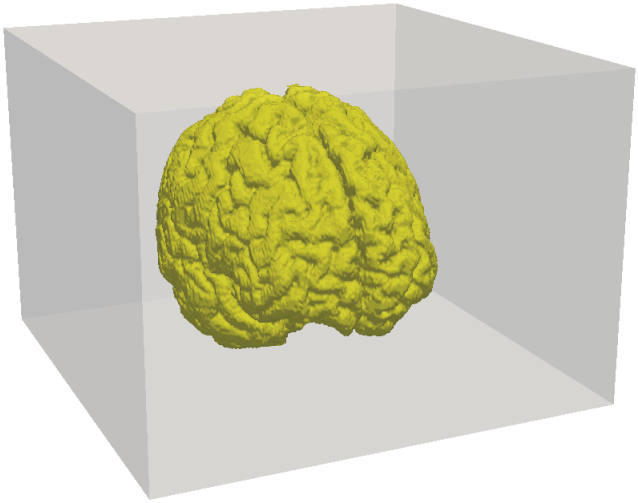
At the end, the results demonstrate that our approach was interestingly applied to 3D head MRI scanned images, and provided accurate redistanced phase functions; as well as the adapted anisotropic mesh. Consequently, the iso-zero surface of the redistanced phase function should give the boundary of the segmented part, even in a smoother way than if it was computed from the original image, by avoiding the pixelation influence. On the other hand, an adapted anisotropic mesh provides a good tool to perform efficient resolutions for medical simulations, by reducing the number of degrees of freedom of the grid. Additionally, from a segmented image, $\widehat{u_{seg}}$, the redistancing-adaptation procedure may require hundreds of iterations to converge to the redistanced function, meaning weeks of computing time. The signed distance computed on the image with processing techniques, $\widehat{u_d}$, may powerfully reduce the computing time.



(a) Segmented head part

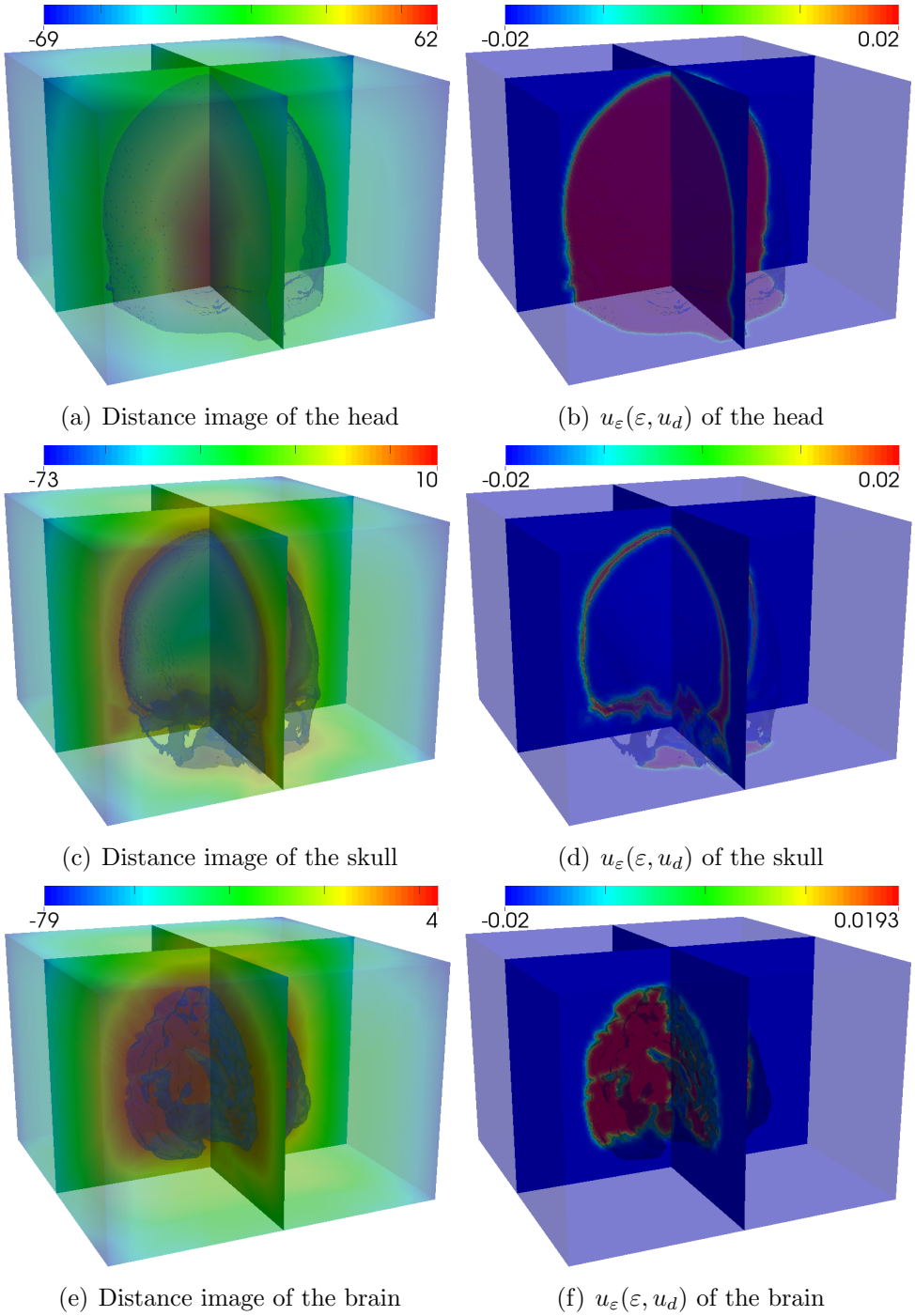


(b) Segmented skull part



(c) Segmented brain part

Figure 3.28: Segmented regions of the head image: whole head, skull and brain.



(a) Distance image of the head (b) $u_\varepsilon(\varepsilon, u_d)$ of the head

(c) Distance image of the skull (d) $u_\varepsilon(\varepsilon, u_d)$ of the skull

(e) Distance image of the brain (f) $u_\varepsilon(\varepsilon, u_d)$ of the brain

Figure 3.29: Image of the computed signed distance from the original image for the three regions: $\widehat{u_{d_{head}}}$, $\widehat{u_{d_{skull}}}$, $\widehat{u_{d_{brain}}}$ and computed hyperbolic tangent function, u_ε, on mesh, from the interpolated distance, u_d.

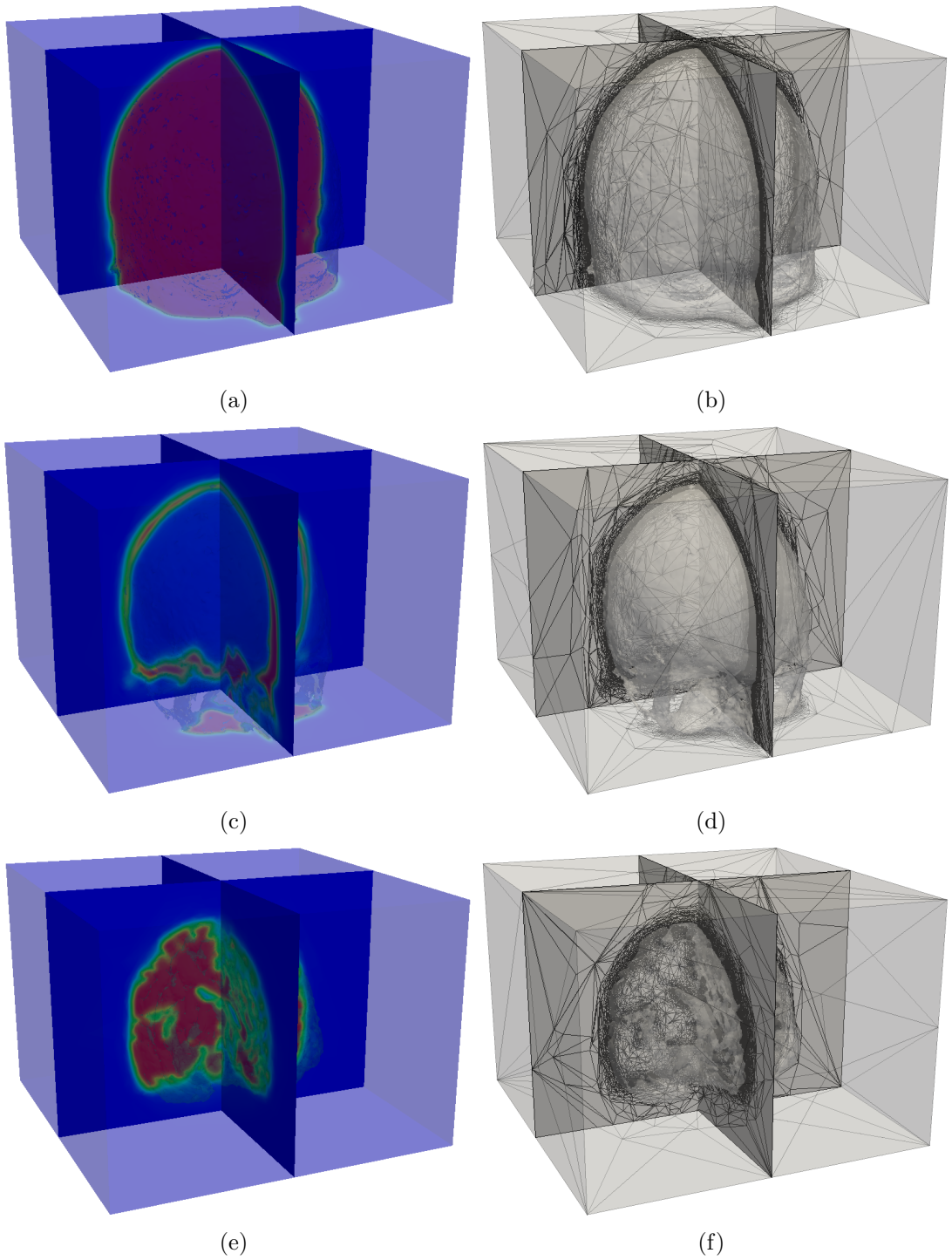
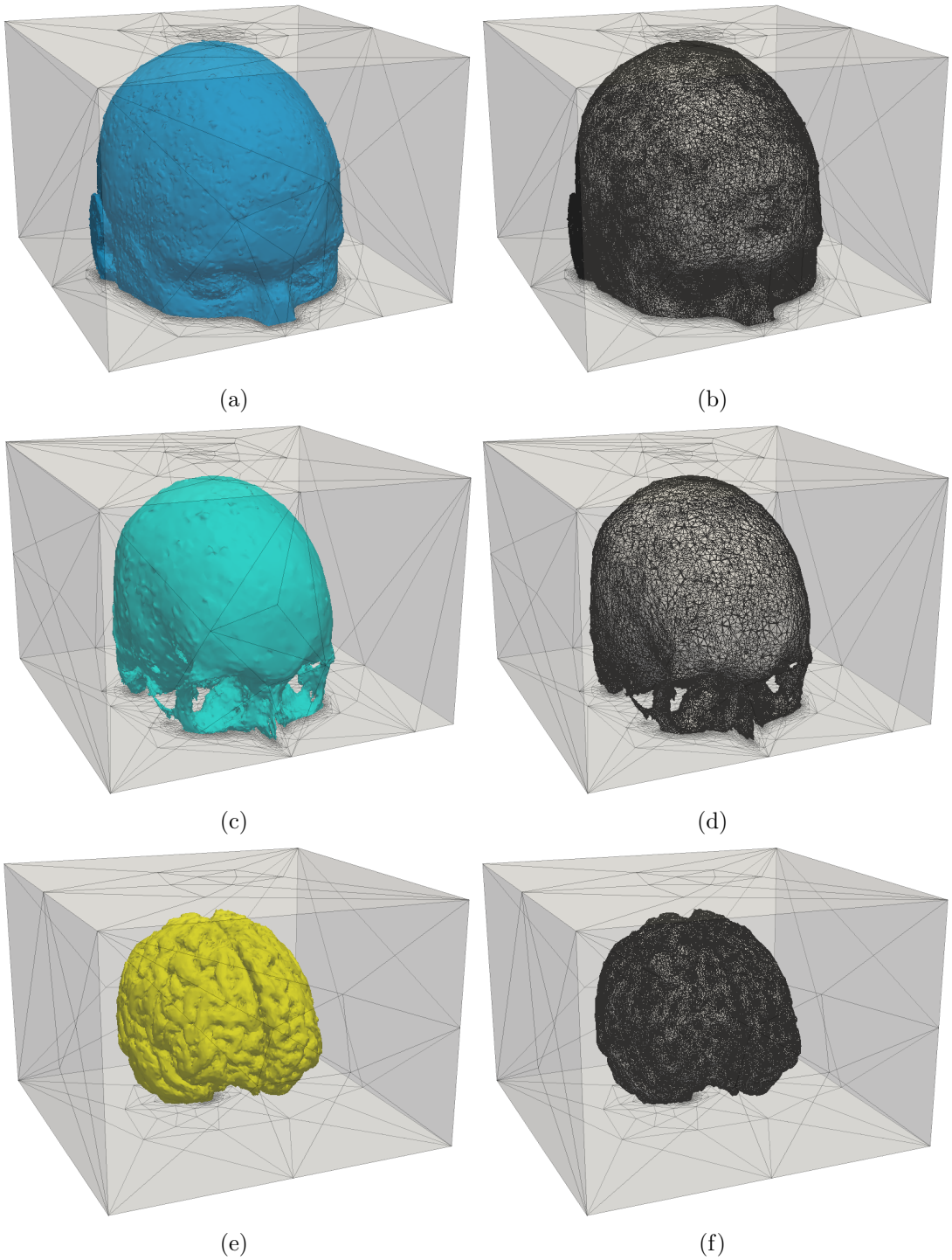


Figure 3.30: Redistanced function u_ε^τ, adapted anisotropic volume mesh for the whole head, the skull and the brain.

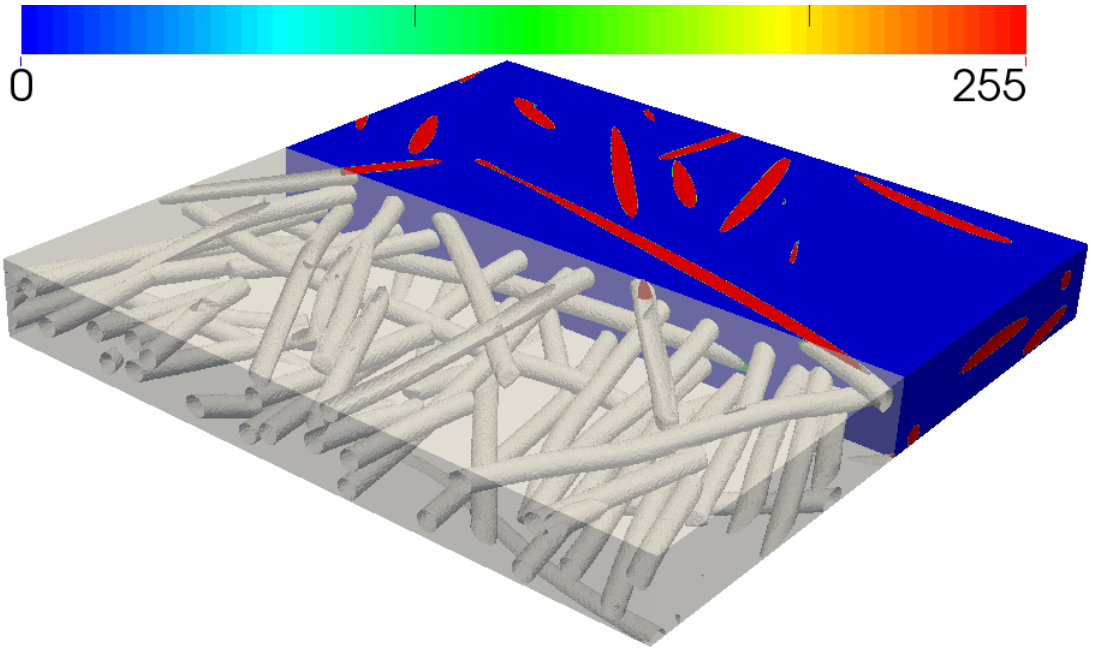


(a) (b) (c) (d) (e) (f)

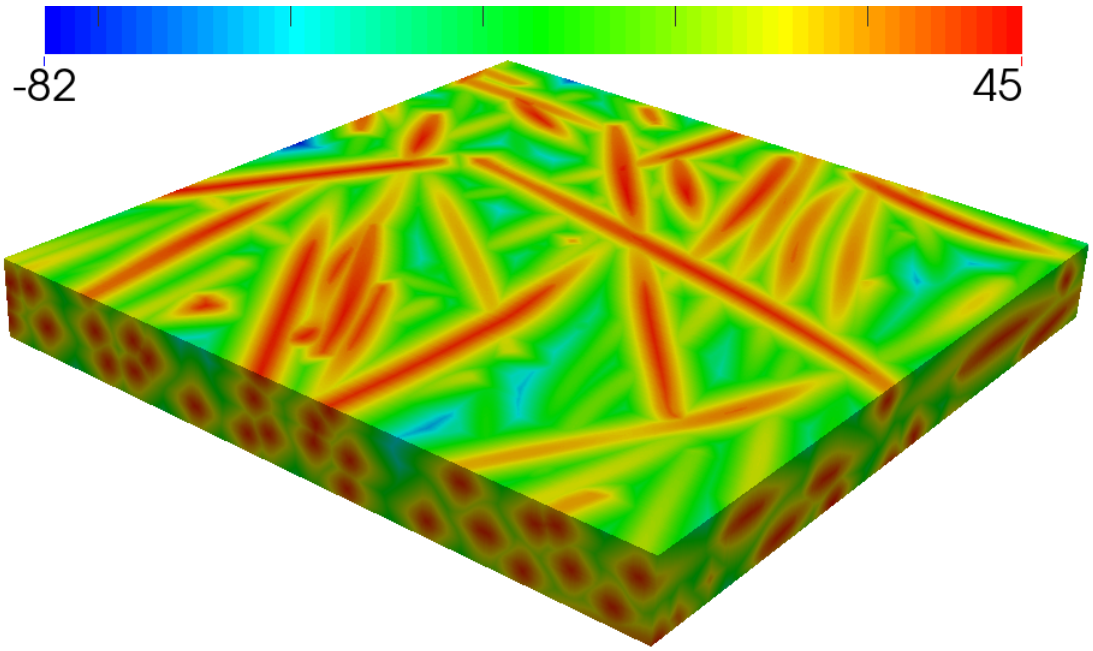
Figure 3.31: Iso-zero surface of u_ε^τ and surface mesh for the whole head, the skull and the brain.

3.4.5 3D Fiber image

This methodology may not only improve medical images, but can also be applied in material science research. For example, let us present the 3D composite image obtained by 3D X-ray micro-tomography [70], which has the dimensions ($\widehat{L} \times \widehat{H} \times \widehat{W} = 900 \times 900 \times 100$) and is illustrated in Figure 3.32.



(a)

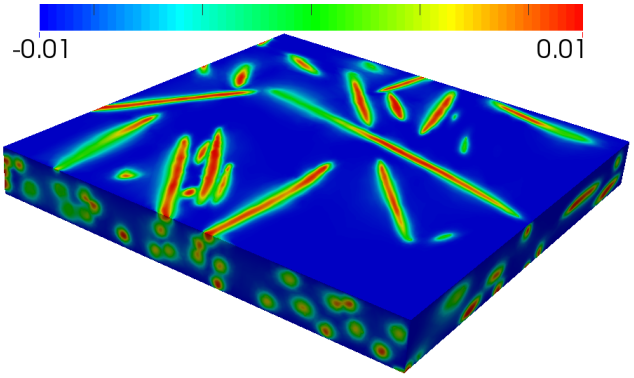


(b)

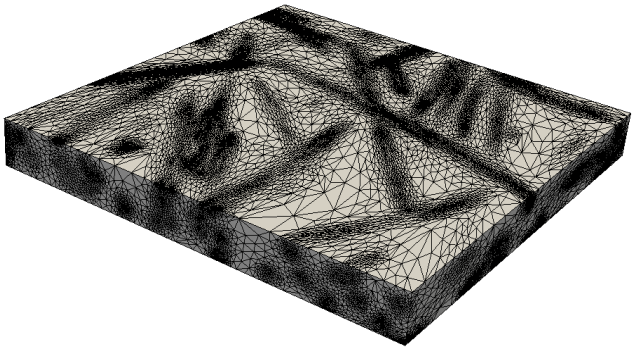
Figure 3.32: Original 3D image and computed signed distance image (obtained using Morph-M).

The goal here is again to create the anisotropic mesh, adapted to the contour of the fibers, which may be applied in many numerical simulations. With the help of Morph-M, the signed distance image, $\widehat{u_d}$, was obtained. It was interpolated on an initial mesh of dimensions $[0, 8.99] \times [0, 8.99] \times [0, 0.99]$, obtaining, the signed distance $u_d(\mathbf{X}^i) = \widehat{u_d}(Voxel^k)/100$. The initial hyperbolic function, $u_\varepsilon^0(\varepsilon, S, \tau = 0) = u_\varepsilon(u_d, \varepsilon)$, was determined and the redistancing-adaptation procedure launched with $(\varepsilon, N) = (0.01, 400000)$. After 25 iterations run on 6 cores and a CPU time of 213.8 minutes, the redistanced function u_ε^τ and the adapted volume anisotropic mesh were obtained and are presented in Figure 3.33.

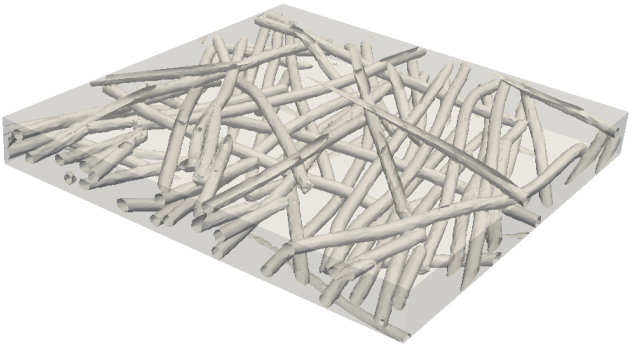
The iso-zero surface of this redistanced function has also been drawn, which accurately represents the geometry of the fibers, and may eventually be used to extract a surface mesh of the fibers.



(a) Redistanced u_ε^τ



(b) Adapted volume mesh



(c) Iso-zero surface of u_ε^τ

(d) Surface mesh of the iso-zero surface of u_ε^τ

Figure 3.33: Illustration of the redistancing-adaptation on an image of a 3D composite.

3.5 Conclusion

In this chapter, a new redistancing modified level-set method was proposed, by building a hyperbolic tangent function with a given thickness ε from an initially discontinuous Heaviside. This redistanced function plays a very important role. Firstly, anisotropic mesh adaptation may benefit from gradient variation, to obtained a more dense and anisotropic mesh around the boundary of object. Secondly, the redistanced function may be used to define the physical parameters everywhere in the computational domain. Finally, this redistancing method builds directly a smoother function, avoiding pixelation traces. The drawback of this method is the number of required iterations to converge both in the mesh and in the function. Fortunately, image processing software, like Morph-M, based on Mathematical Morphology techniques, provide several interesting image processing operations, such as segmentation, filtering or distance computation. The construction of an image image close to the function to be built may reduce the number of iterations and save computational time. The introduced redistancing-adaptation method coupled to Morph-M may be used in medical and material simulations, providing the constructed mesh directly based on the 3D image.

3.6 Résumé français

Dans ce chapitre, nous proposons une nouvelle méthode de réinitialisation d'une fonction type "level-set" en construisant une fonction tangente hyperbolique avec d'une épaisseur ε à partir d'une fonction Heaviside discontinue. Cette fonction distance joue un rôle important dans cette méthode. Premièrement, l'adaptation de maillage anisotrope se base sur la variation de cette fonction distance. Ainsi, le maillage est plus dense autour de la frontière de l'objet. Deuxièmement, les fonctions réinitialisées peuvent être utilisés pour définir les paramètres physiques du domaine. Enfin, la méthode de réinitialisation permet de construire des fonctions régulières, évitant la sur-pixelisation. Cependant, de nombreuses itérations sont nécessaires à la convergence du maillage et de la fonction régularisée. Pour pallier ce probl eme, de nombreux logiciels de traitement d'image comme Morph-M, basé sur des techniques de morphologie mathématique, permettent de nombreuses opérations comme la segmentation, le filtrage ou le calcul de la distance. La construction de la fonction distance permet de réduire le nombre d'itérations et le temps de calcul. Cette méthode, couplée à Morph-M peut être utilisée dans le domaine médical et en physique des matériaux, par construction d'un maillage sur les données d'une image.

Chapter 4

A monolithic approach for multiphase computational flow simulation

Contents

4.1 Introduction

Multiphase flow studies are important in several industrial applications like, for example, nuclear reactors, automotive manufacturing, aircraft design or biological flows. In this context, the level-set method is often used to represent, in an implicit way, the geometries and properties of the different phases. In addition, the previously presented anisotropic mesh adaptation technique may help capture details, to perfectly describe the complex issues with the smallest computational cost and computing time.

This chapter presents the numerical tools coupled to our image mesher coupled and used to model and solve multiphase flows. Indeed, we have proposed an unique constitutive set of equations, solved in the entire computational domain, combined with the variation of the physical parameters, defined from the different phases' properties. The overall computation procedure consists mainly in the resolution of the Navier-Stokes equations using this monolithic approach and an Eulerian framework, solved with a VMS (Variational MultiScale) finite element method [61, 71]. Some numerical simulations based on real images will be illustrated.

4.2 Navier-Stokes equations

Firstly, let us consider $\Omega \subset \mathbb{R}^d$ as the whole computational domain, where d is the space dimension, and $\partial\Omega$ is the boundary of Ω. To detail the multiphase flow context, we consider the simplest case, of a fluid interacting with a solid. The fluid and solid domains are respectively noted as Ω_f, Ω_s, and the interface between the two domains is, Γ_{im}.

$$\Omega_f \cap \Omega_s = \Gamma_{im} \text{ and } \Omega_f \cup \Omega_s = \Omega \tag{4.1}$$

The purpose is to obtain the velocity and pressure fields, $\forall\ x \in \Omega$. The classical incompressible Navier-Stokes equations for computing the flow dynamics of a Newtonian fluid in Ω_f are as follows: find $(\mathbf{v}, p)$ such that

$$\begin{cases} \rho_f(\dfrac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}) - \nabla \cdot \boldsymbol{\sigma} = \mathbf{f} & \text{in } \Omega_f, t > 0 \\ \nabla \cdot \mathbf{v} = 0 & \text{in } \Omega_f, t > 0 \end{cases} \tag{4.2}$$

where the velocity $\mathbf{v}(\mathbf{x}, t)$ is time dependent, ρ_f is the fluid density and the Cauchy stress tensor for a Newtonian fluid is given by:

$$\boldsymbol{\sigma} = 2\eta_f \boldsymbol{\varepsilon}(\mathbf{v}) - p\mathbf{I}_d \tag{4.3}$$

$\mathbf{I}_d$ is the d-dimension identity tensor, and η_f is the fluid viscosity. Let us introduce the boundary and initial conditions in following equations:

$$\begin{cases} \mathbf{v} = \mathbf{v}_{\Gamma,f} & \text{on } \partial\Omega_f \setminus \Gamma_{im}, t > 0 \\ \mathbf{v} = \mathbf{v}_{im} & \text{on } \Gamma_{im}, t > 0 \\ \boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{t}_{im} & \text{on } \Gamma_{im}, t > 0 \\ \mathbf{v}(., 0) = \mathbf{v}_0 & \text{in } \Omega_f \end{cases} \tag{4.4}$$

where $\mathbf{v}_{\Gamma,f}$ is the velocity boundary condition, $\mathbf{v}_{im}$ is the velocity at the fluidsolid interface, and the boundary of the immersed solid is Γ_{im}. $\mathbf{n}$ is the outward normal at the solid surface, $\mathbf{t}_{im}$ is designed as the normal stress on this boundary, and $\mathbf{v}_0$ is a given initial condition on the velocity field.

The solid domain is considered as a rigid body, the classical incompressible Navier-Stokes equation are modified by adding an additional rigidity constraint as:

$$\begin{cases} \rho_s(\dfrac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}) - \nabla\cdot\boldsymbol{\sigma} = \mathbf{f} & \text{in } \Omega_s, t>0 \\ \nabla\cdot\mathbf{v} = 0 & \text{in } \Omega_s, t>0 \\ \boldsymbol{\varepsilon}_s(\mathbf{v}) = \mathbf{0} & \text{in } \Omega_s, t>0 \end{cases} \tag{4.5}$$

where ρ_s is the solid density. For the rigid body motion, the deformation of the rigid object is zero, the deformation-rate tensor being $\boldsymbol{\varepsilon}(\mathbf{v}) = 0$. We consider here a rigidity constraint, using $\boldsymbol{\tau}_s$, and where this extra stress will represent the presence of the structure in the fluid domain. Hence, one may establish

$$\boldsymbol{\sigma} = \boldsymbol{\tau}_s - p\mathbf{I}_d \tag{4.6}$$

The boundary and initial conditions are given below, with the same meaning has the ones described for the fluid domain.

$$\begin{cases} \mathbf{v} = \mathbf{v}_{\Gamma,s} & \text{on } \partial\Omega_s \setminus \Gamma_{im}, t>0 \\ \mathbf{v} = \mathbf{v}_{im} & \text{on } \Gamma_{im}, t>0 \\ \boldsymbol{\sigma}\cdot\mathbf{n} = -\mathbf{t}_{im} & \text{on } \Gamma_{im}, t>0 \\ \mathbf{v}(\mathbf{x},0) = \mathbf{v}_0 & \text{in } \Omega_s \end{cases} \tag{4.7}$$

4.3 Monolithic approach and Eulerian formulation

4.3.1 Full Eulerian formulation

Different approaches exist to handle a FSI or a multiphase flow problem, which are often distinguished between Lagrangian, Eulerian or ALE (Arbitrary Lagrangian-Eulerian) formulations. We briefly recall these three approaches.

A **Lagrangian** formulation describes the motion of a body starting from a known reference, and each particle of the body is attached to this reference system. When using the finite element method, each node of the mesh is coincident with its corresponding material particle through the entire deformation and displacement procedure. More generally, this formulation is used to track moving boundaries of the solid object and it often requires mesh adaptation because of the domain's distortion, otherwise we are lead to inaccurate results.

An **Eulerian** formulation in a finite element method context requires a fixed mesh in the whole computed space, defining a constant computational domain over the time. It is usually used for tracking the fluid motion in the space as time evolves. However, it may be more difficult to establish it to study solid deformation and

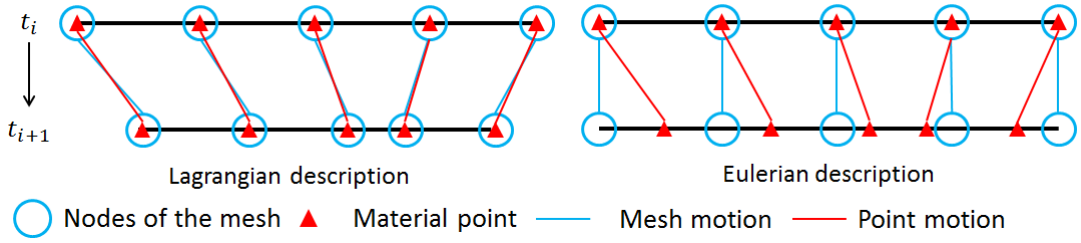


Figure 4.1: Schematic illustration of Lagrangian and Eulerian descriptions in an 1D mesh and corresponding particle motion.

displacement, when several material particles leave at once their formerly occurred state, leading to a loss of information.

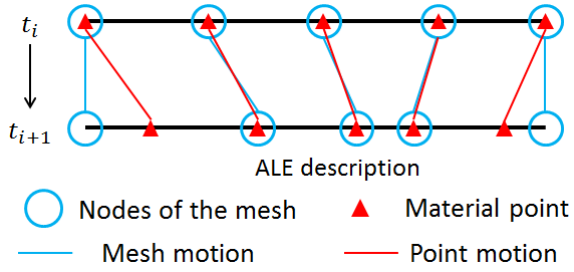


Figure 4.2: *ALE* description in a 1D mesh and material particle motion description.

An **ALE** formulation [72] combines the advantages of both Lagrangian and Eulerian formulations, being neither only fixed in space (Eulerian) or only attached to the material (Lagrangian). The idea of this mixed formulation in Fluid-Structure Interaction is often to use the Lagrangian formulation for the structure and an Eulerian formulation for the incompressible fluid.

As introduced in the previous chapter, modified level-set functions may be used to represent the different phases, coupled to anisotropic mesh adaptation, and therefore adequate for an Eulerian formulation to represent and capture the motion, for both solid and liquid. That is the chosen approach in this work, to present both the fluid and structural domains, throughout flow computations.

4.3.2 Monolithic approach

To solve multiphase flows, and in particular fluid-structure interaction problems, there are also two main techniques: **partitioned** and **monolithic**.

The **partitioned** approach uses a specific solver for each domain. The global problem is divided into smaller parts and solved independently. However, the difficulty happens when the exchanging information for one solver to the other after each iteration and, to overcome these difficulties, coupling algorithms have been proposed [73]. Sometimes, instability conditions on this coupling may lead to an information loss and adds also a certain computation cost.

The **monolithic** approach is based on one single grid or set of equations to describe the different phases' mechanics and, unlike partitioned approaches where

different solvers handle different objects, only one global resolution is made. Some examples to handle FSI problems using this approach are presented in [74, 75].

In this paper, we have used a monolithic approach in an Eulerian formulation to handle multiphase flow problems. Different objects are represented by implicit continuous functions in the mesh, like the one described and computed in the previous chapter. Multiphase Navier-Stokes equations are established coupling the previously ones for fluid and rigid body motions, with specific boundary conditions. The set of equations to be solved may be presented as:

$$\begin{cases} \rho\dfrac{\partial \mathbf{v}}{\partial t} + \rho\mathbf{v}\cdot\nabla\mathbf{v} - 2\eta_f\nabla\cdot\boldsymbol{\varepsilon}(\mathbf{v}) - \nabla\cdot\boldsymbol{\tau} + \nabla p = f & \text{in } \Omega_f, t>0 \\ \nabla\cdot\mathbf{v} = 0 & \text{in } \Omega, t>0 \\ \boldsymbol{\varepsilon}_s(\mathbf{v}) = \mathbf{0} & \text{in } \Omega, t>0 \\ \mathbf{v} = \mathbf{v}_\Gamma & \text{on } \partial\Omega, t>0 \\ \mathbf{v}(\mathbf{x},0) = \mathbf{v}_0 & \text{in } \Omega \end{cases} \tag{4.8}$$

where $\mathbf{v} = \mathbf{v}_{\Gamma,f}$ on $\partial\Omega_f \cap \partial\Omega$ and $\mathbf{v} = \mathbf{v}_{\Gamma,s}$ on $\partial\Omega_s \cap \partial\Omega$, ρ, η, $\boldsymbol{\varepsilon}$ and $\boldsymbol{\tau}$ are the density, dynamic viscosity, deformation-rate and stress tensor fields on the whole computational domain.

4.3.3 Mixture laws

The previously presented multiphase Navier-Stokes equations require the definition of the mechanical properties in the whole domain. On the other hand, the geometry of each object or phase in the image is defined at the mesh nodes using a modified level-set function, such as the hyperbolic tangent one, u_ε. This distribution allows us to compute the mechanical properties with, for example, mixing laws.

Let us define a discontinuous Heaviside function, computed from the level-set one and noted $H(u_\varepsilon)$, as:

$$H(u_\varepsilon) = \begin{cases} 1 & \text{if } u_\varepsilon \geq 0 \\ 0 & \text{in } u_\varepsilon < 0 \end{cases} \tag{4.9}$$

In multiphase problems, several physical phenomena happen at the interface between the different phases and, to model them one often prefers continuity of the properties across this interface even if, in reality, they are discontinuous. Thus, a smooth Heaviside function, $H_\varepsilon(u_\varepsilon)$, is built as follows:

$$H_\varepsilon(u_\varepsilon) = \begin{cases} 1 & \text{if } u_\varepsilon > \varepsilon \\ \dfrac{1}{2}\left(1 + \dfrac{u_\varepsilon}{\varepsilon} + \dfrac{1}{\pi}\sin\left(\dfrac{\pi u_\varepsilon}{\varepsilon}\right)\right) & \text{if } |u_\varepsilon| \leq \varepsilon \\ 0 & \text{if } u_\varepsilon < -\varepsilon \end{cases} \tag{4.10}$$

where ε is the wanted "interface" thickness. Using this smooth function, properties like density and viscosity are determined through:

$$\begin{cases} \rho = \rho_s H_\varepsilon(u_\varepsilon) + \rho_f \left(1 - H_\varepsilon(u_\varepsilon)\right) \\ \eta = \eta_s H_\varepsilon(u_\varepsilon) + \eta_f \left(1 - H_\varepsilon(u_\varepsilon)\right) \\ \boldsymbol{\varepsilon}_s = H_\varepsilon(u_\varepsilon)\boldsymbol{\varepsilon} \\ \boldsymbol{\tau} = H_\varepsilon(u_\varepsilon)\boldsymbol{\tau}_s \end{cases} \tag{4.11}$$

4.4 Variation MultiScale method

In this section, we present the resolution of the multiphase Navier-Stokes problem, through the Variation MultiScale finite element method. Firstly, let us define the following functional spaces:

$$\begin{cases} \mathcal{V} = \left\{ \mathbf{v}, \mathbf{v} \in \left(H^1(\Omega)\right)^d \right\} \\ \mathcal{P} = \left\{ p, p \in L^2(\Omega) \right\} \\ \mathcal{T} = \left\{ \boldsymbol{\tau}, \boldsymbol{\tau} \in L^2(\Omega)^{n \times n} \right\} \end{cases} \tag{4.12}$$

Then, the corresponding test functions will be denoted $\mathbf{w} \in \mathcal{V}^0 = H_0^1(\Omega)^d$, $q \in \mathcal{P}^0$ and $\boldsymbol{\xi} \in \mathcal{T}^0$. The strong form of Equations (4.8) is associated to a standard weak one by integrating over the whole computational domain. It may be written as:

$$\begin{cases} \rho \left(\dfrac{\partial \mathbf{v}}{\partial t}, \mathbf{w} \right) + \rho \left(\mathbf{v} \cdot \nabla \mathbf{v}, \mathbf{w} \right) - \left(p, \nabla \cdot \mathbf{w} \right) + \left(2\eta \boldsymbol{\varepsilon}\left(\mathbf{v}\right), \boldsymbol{\varepsilon}\left(\mathbf{w}\right) \right) + \left(\boldsymbol{\tau}, \boldsymbol{\varepsilon}_s\left(\mathbf{w}\right) \right) = \left(\mathbf{f}, \mathbf{w} \right) \\ \left(q, \nabla \cdot \mathbf{v} \right) = 0 \\ - \left(\boldsymbol{\xi}, \boldsymbol{\varepsilon}_s\left(\mathbf{v}\right) \right) = 0 \end{cases} \tag{4.13}$$

where $(\mathbf{v}, p, \boldsymbol{\tau}) \in (\mathcal{V}, \mathcal{P}, \mathcal{T})$ and the test functions are $(\mathbf{w}, q, \boldsymbol{\xi}) \in (\mathcal{V}^0, \mathcal{P}^0, \mathcal{T}^0)$. Based on the mesh discretization $\mathcal{K}$ of Ω, velocity $\mathbf{v}$, pressure p and stress fields $\boldsymbol{\tau}$ may be discretized using the finite dimensional spaces, leading to the interpolated $\mathbf{v}_h$, p_h and $\boldsymbol{\tau}_h$. However, resolution of the Equations (4.13) using the standard Galerkin finite element method may fail and lead to non-physical spurious oscillations. The first reason is related to the presence of convection terms, and the second one concerns the loss of stability because the mixed finite element formulation does not satisfy the inf-sup (Ladyzhenskaya-Brezzi-Babuška) condition. To stabilize this discrete formulation, and inspired from the previously presented $SUPG$ and RFB methods, the alternative VMS one was implemented and is described in this section [36].

The basic idea behind the VMS method is to use the decomposition of the velocity, pressure and stress fields into coarse-scale and fine-scale values. Then, one solves the fine-scale and replaces its effect in the coarse-scale. Let us denote the

element interiors and boundaries by Ω' and Γ'. The velocity $\mathbf{v}$, pressure p and stress $\boldsymbol{\tau}$ fields can be approximated as:

$$\begin{cases} \mathbf{v} = \mathbf{v}_h + \mathbf{v}' & \in \mathcal{V}_h \oplus \mathcal{V}' \\ p = p_h + p' & \in \mathcal{P}_h \oplus \mathcal{P}' \\ \boldsymbol{\tau} = \boldsymbol{\tau}_h + \boldsymbol{\tau}' & \in \mathcal{T}_h \oplus \mathcal{T}' \end{cases} \tag{4.14}$$

where $(\mathbf{v}_h + \mathbf{v}', p_h + p', \boldsymbol{\tau}_h + \boldsymbol{\tau}') \in (\mathcal{V}_h \oplus \mathcal{V}' \times \mathcal{P}_h \oplus \mathcal{P}' \times \mathcal{T}_h \oplus \mathcal{T}')$. For the fine-scale, their values vanish over the element boundaries:

$$\begin{cases} \mathbf{v}' = \mathbf{0} \\ p' = 0 \qquad \text{on } \Gamma' \\ \boldsymbol{\tau}' = \mathbf{0} \end{cases} \tag{4.15}$$

Assuming that the velocity, pressure and stress at the fine-scale are piecewise polynomial and continuous in space, and the test functions may have the same decomposition as $\mathbf{w} = \mathbf{w}_h + \mathbf{w}' \in \mathcal{V}_h^0 \oplus \mathcal{V}^{',0}, q = q_h + q' \in \mathcal{P}_h^0 \oplus \mathcal{P}^{',0}, \boldsymbol{\xi} = \boldsymbol{\xi}_h + \boldsymbol{\xi}' \in \mathcal{T}_h^0 + \mathcal{T}^{',0}$. The mixed-finite element approximations of Equation (4.13) can be written

$$\begin{cases} \rho\left(\dfrac{\partial(\mathbf{v}_h + \mathbf{v}')}{\partial t}, \mathbf{w}_h + \mathbf{w}'\right) + \rho\left((\mathbf{v}_h + \mathbf{v}') \cdot \nabla(\mathbf{v}_h + \mathbf{v}'), \mathbf{w}_h + \mathbf{w}'\right) - \left(p_h + p', \nabla \cdot (\mathbf{w}_h + \mathbf{w}')\right) \\ \quad + \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h + \mathbf{v}'\right), \boldsymbol{\varepsilon}\left(\mathbf{w}_h + \mathbf{w}'\right)\right) + \left(\boldsymbol{\tau}_h + \boldsymbol{\tau}', \boldsymbol{\varepsilon}_s\left(\mathbf{w}_h + \mathbf{w}'\right)\right) = \left(\mathbf{f}, \mathbf{w}_h + \mathbf{w}'\right) \\ \\ \left(q_h + q', \nabla \cdot (\mathbf{v}_h + \mathbf{v}')\right) = 0 \\ \\ -\left(\boldsymbol{\xi}_h + \boldsymbol{\xi}', \boldsymbol{\varepsilon}_s\left(\mathbf{v}_h + \mathbf{v}'\right)\right)_{\Omega_s} = 0 \end{cases} \tag{4.16}$$

Therefore, Equation (4.16) may be decomposed in two sub-ones:
(1) the coarse-scale problem

$$\begin{cases} \rho\left(\dfrac{\partial(\mathbf{v}_h + \mathbf{v}')}{\partial t}, \mathbf{w}_h\right) + \rho\left(\mathbf{v}_h \cdot \nabla(\mathbf{v}_h + \mathbf{v}'), \mathbf{w}_h\right) - \left(p_h + p', \nabla \cdot \mathbf{w}_h\right) \\ \quad + \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h + \mathbf{v}'\right), \boldsymbol{\varepsilon}\left(\mathbf{w}_h\right)\right) + \left(\boldsymbol{\tau}_h + \boldsymbol{\tau}', \boldsymbol{\varepsilon}_s\left(\mathbf{w}_h\right)\right) = \left(\mathbf{f}, \mathbf{w}_h\right) \\ \\ \left(q_h, \nabla(\mathbf{v}_h + \mathbf{v}')\right) = 0 \\ \\ -\left(\boldsymbol{\xi_h}, \boldsymbol{\varepsilon}_s\left(\mathbf{v}_h + \mathbf{v}'\right)\right)_{\Omega_s} = 0 \end{cases} \tag{4.17}$$

(2) the fine-scale problem

$$\begin{cases} \rho\left(\dfrac{\partial(\mathbf{v}_h+\mathbf{v}')}{\partial t},\mathbf{w}'\right)+\rho\left(\mathbf{v}_h\cdot\nabla(\mathbf{v}_h+\mathbf{v}'),\mathbf{w}'\right)-\left(p_h+p',\nabla\cdot\mathbf{w}'\right) \\ \quad+\left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h+\mathbf{v}'\right),\boldsymbol{\varepsilon}\left(\mathbf{w}'\right)\right)+\left(\boldsymbol{\tau}_h+\boldsymbol{\tau}',\boldsymbol{\varepsilon}_s\left(\mathbf{w}'\right)\right)=\left(\mathbf{f},\mathbf{w}'\right) \\ \left(q',\nabla\cdot\left(\mathbf{v}_h+\mathbf{v}'\right)\right)=0 \\ -\left(\boldsymbol{\xi}',\boldsymbol{\varepsilon}_s\left(\mathbf{v}_h+\mathbf{v}'\right)\right)=0 \end{cases} \tag{4.18}$$

Three important remarks and hypothesis need to be considered:

- when a linear interpolation is used, the second derivatives vanish as well as all the terms involved is the integrals over the element interior boundaries;
- the fine-scale is not stored over the time, by considering the problem as quasi-static for this scale;
- the convective velocity in the the non-linear term may be approximated using only the coarse-scale values, in such a way that

$$\left(\mathbf{v}_h+\mathbf{v}'\right)\nabla\cdot\left(\mathbf{v}_h+\mathbf{v}'\right)\approx\mathbf{v}_h\cdot\nabla\left(\mathbf{v}_h+\mathbf{v}'\right) \tag{4.19}$$

The equations for the coarse scales are obtained taking the fine-scale test functions equal to zero. With the previously described assumptions, Equation (4.17) may be rewritten as:

$$\begin{cases} \rho\left(\dfrac{\partial\mathbf{v}_h}{\partial t},\mathbf{w}_h\right)+\rho\left(\mathbf{v}_h\cdot\nabla\mathbf{v}_h,\mathbf{w}_h\right)-\left(p_h+p',\nabla\cdot\mathbf{w}_h\right)+\left(2\eta\boldsymbol{\varepsilon}(\mathbf{v}_h),\boldsymbol{\varepsilon}(\mathbf{w}_h)\right)+\left(\boldsymbol{\tau}_h+\boldsymbol{\tau}',\boldsymbol{\varepsilon}_s(\mathbf{w}_h)\right) \\ \quad+\displaystyle\sum_{K\in\mathcal{K}}\left(\mathbf{v}',-\rho\mathbf{v}_h\cdot\nabla\mathbf{w}_h-\nabla\cdot\left(2\eta\boldsymbol{\varepsilon}(\mathbf{w}_h)\right)\right)_K=\left(\mathbf{f},\mathbf{w}_h\right) \\ \left(q_h,\nabla\cdot\mathbf{v}_h\right)-\displaystyle\sum_{K\in\mathcal{K}}\left(\mathbf{v}',\nabla q_h\right)_K=0 \\ -\left(\boldsymbol{\varepsilon}_s(\mathbf{v}_h),\boldsymbol{\xi}_h\right)+\displaystyle\sum_{K\in\mathcal{K}}\left(\mathbf{v}',\chi_s\nabla\cdot\boldsymbol{\xi}_h\right)_K=0 \end{cases} \tag{4.20}$$

for all $(\mathbf{w}_h,q_h,\boldsymbol{\xi}_h)\in\mathcal{V}_h^0\times\mathcal{P}_h^0\times\mathcal{T}_h^0$, where $\sum_{K\in\mathcal{K}}$ means the summation over all the elements of the finite set $\mathcal{K}$. Let us consider now the fine-scale problem with $(\mathbf{w}_h,q_h,\boldsymbol{\xi}_h)=(\mathbf{0},0,\mathbf{0})$. In [76], the finite element residuals are defined as:

$$\begin{cases} \mathcal{R}_v=\mathbf{f}-\rho\dfrac{\partial\mathbf{v}_h}{\partial t}-\rho\mathbf{v}_h\cdot\nabla\mathbf{v}_h-\nabla p_h+\chi_s\nabla\cdot\boldsymbol{\tau}_h+\nabla\cdot\left(2\eta\boldsymbol{\varepsilon}(\mathbf{v}_h)\right) \\ \mathcal{R}_p=-\nabla\cdot\mathbf{v}_h \\ \mathcal{R}_{\boldsymbol{\tau}}=\boldsymbol{\varepsilon}_s\left(\mathbf{v}_h\right) \end{cases} \tag{4.21}$$

Using a fine-scale definition like the one presented previously for the RFB method, $(\mathbf{v}', p', \boldsymbol{\tau}')$ may be written as bubble functions as:

$$\begin{cases} \mathbf{v}' = \sum_{K \in \mathcal{K}} b_{\mathbf{v}} \mathbf{v}'_K \\ p' = \sum_{K \in \mathcal{K}} b_p p'_K \\ \boldsymbol{\tau}' = \sum_{K \in \mathcal{K}} b_{\boldsymbol{\tau}} \boldsymbol{\tau}'_K \end{cases} \tag{4.22}$$

where $b_{\mathbf{v}}, b_p, b_{\boldsymbol{\tau}}$ are the bubble functions [63], and $\mathbf{v}'_K, p'_K, \boldsymbol{\tau}'_K$ are the fine-scale values on element K. Additionally, $\mathbf{v}'_K, p'_K, \boldsymbol{\tau}'_K$ may also be written as functions of $(a_{\mathbf{v}}, a_p, a_{\boldsymbol{\tau}})$ and $(\mathbf{v}_h, p_h, \boldsymbol{\tau}_h)$, where $a_{\mathbf{v}}, a_p, a_{\boldsymbol{\tau}}$ are stabilization parameters, computed within each element. More details are presented in [77, 78]. Finally,

$$\begin{cases} \mathbf{v}' = \sum_{K \in \mathcal{K}} a_{\mathbf{v}}(\mathcal{R}_v) \\ p' = \sum_{K \in \mathcal{K}} a_p(\mathcal{R}_p) \\ \boldsymbol{\tau}' = \sum_{K \in \mathcal{K}} a_{\boldsymbol{\tau}}(\mathcal{R}_{\boldsymbol{\tau}}) \end{cases} \tag{4.23}$$

This provides us the expressions for the subscales, $\mathbf{v}'$, p', $\boldsymbol{\tau}'$. Then, we go back to the coarse-scale problem, by substituting this fine-scale effects $\mathbf{v}'$, p', $\boldsymbol{\tau}'$. Applying the integration by parts, we get:

$$
\left\{
\begin{array}{l}
\underbrace{\rho\left(\dfrac{\partial \mathbf{v}_h}{\partial t}, \mathbf{w}_h\right) + \rho\left(\mathbf{v}_h \cdot \nabla \mathbf{v}_h, \mathbf{w}_h\right) - \left(p_h, \nabla \cdot \mathbf{w}_h\right) + \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h\right), \boldsymbol{\varepsilon}\left(\mathbf{w}_h\right)\right) + \left(\boldsymbol{\tau}_h, \boldsymbol{\varepsilon}_s\left(\mathbf{w}_h\right)\right)}_{\text{Galerkin terms}} \\
\quad + \underbrace{\displaystyle\sum_{K\in\mathcal{K}} a_{\mathbf{v}}\left(\rho\frac{\partial \mathbf{v}_h}{\partial t} + \rho\mathbf{v}_h \cdot \nabla \mathbf{v}_h + \nabla p_h - \chi_s \nabla \cdot \boldsymbol{\tau}_h - \nabla \cdot \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h\right)\right), \rho\mathbf{v}_h \cdot \nabla \mathbf{w}_h + \nabla \cdot \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{w}_h\right)\right)\right)_K}_{\text{Upwind stabilization terms}} \\
\quad + \underbrace{\displaystyle\sum_{K\in\mathcal{K}} a_p\left(\nabla \cdot \mathbf{v}_h, \nabla \cdot \mathbf{w}_h\right)_K}_{\text{grad-div stabilization term}} \\
\quad + \displaystyle\sum_{K\in\mathcal{K}} a_{\boldsymbol{\tau}}\left(\boldsymbol{\varepsilon}\left(\mathbf{v}_h\right), \boldsymbol{\varepsilon}_s\left(\mathbf{w}_h\right)\right)_K \\
= \left(\mathbf{f}, \mathbf{w}_h\right) + \displaystyle\sum_{K\in\mathcal{K}} a_{\mathbf{v}}\left(\mathbf{f}, \rho\mathbf{v}_h \cdot \nabla \mathbf{w}_h + 2\eta\nabla \cdot \boldsymbol{\varepsilon}_s\left(\mathbf{w}_h\right)\right)_K \\
\\
\underbrace{\left(q_h, \nabla \cdot \mathbf{v}_h\right) + \displaystyle\sum_{K\in\mathcal{K}} a_{\mathbf{v}}\left(\rho\frac{\partial \mathbf{v}_h}{\partial t} + \rho\mathbf{v}_h \cdot \nabla \mathbf{v}_h + \nabla p_h - \chi_s \nabla \cdot \boldsymbol{\tau}_h - \nabla \cdot \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h\right)\right), \nabla q_h\right)_K}_{\text{Pressure stabilization terms}} \\
= \displaystyle\sum_{K\in\mathcal{K}} a_{\mathbf{v}}\left(\mathbf{f}, \nabla q_h\right)_K \\
\\
\underbrace{-\left(\boldsymbol{\xi}_h, \boldsymbol{\varepsilon}_s\left(\mathbf{v}_h\right)\right) + \displaystyle\sum_{K\in\mathcal{K}} a_{\mathbf{v}}\left(\rho\frac{\partial \mathbf{v}_h}{\partial t} + \rho\mathbf{v}_h \cdot \nabla \mathbf{v}_h + \nabla p_h - \chi_s \nabla \cdot \boldsymbol{\tau}_h - \nabla \cdot \left(2\eta\boldsymbol{\varepsilon}\left(\mathbf{v}_h\right)\right), -\chi_s \nabla \cdot \xi_h\right)_K}_{\text{Stress stabilization terms}} \\
= \displaystyle\sum_{K\in\mathcal{K}} a_{\mathbf{v}}\left(\mathbf{f}, -\chi_s \nabla \cdot \xi_h\right)_K
\end{array}
\right. \tag{4.24}
$$

In this formulation, all the added terms provide the necessary stabilization for convection-dominated problems, and also circumvent the inf-sup condition.

4.5 Convective level-set method

Moving interfaces on multiphase flows are classical physical phenomena. In the multiphase formulation previously presented, this motion is not explicit, but done through the evolution of an implicit function. Many methods have been used to track interfaces with such type of function, like the *MAC* (Marker And Cell) method [79], the Volume Of Fluid method [80], or the Level-set method [12, 81, 57]. In this section, we have used a modified Level-set approach coupled to a stabilized convective redistancing equation to track the interface motion [82, 83].

In the following, we explain that this can be done as an extension of the previously presented redistancing method. Let us recall that, each object being represented by the hyperbolic phase function $u_\varepsilon(\varepsilon, u_d)$, function of the signed distance function u_d to the object's boundary Γ and of a given thickness ε, such that :

$$\begin{cases} u_\varepsilon(\varepsilon, u_d) = \varepsilon \tanh(\dfrac{u_d(x,\Gamma)}{\varepsilon}) \\ ||\nabla u_\varepsilon||_2 = 1 - (\dfrac{u_\varepsilon}{\varepsilon})^2 = g(u_\varepsilon) \end{cases} \tag{4.25}$$

The velocity field $\mathbf{v}$ is obtained by solving the multiphase Navier-Stokes equations, and the motion of the boundary may be obtained by solving the following advection equation:

$$\begin{cases} \dfrac{\partial u_\varepsilon}{\partial t} + \mathbf{v} \cdot \nabla u_\varepsilon = 0 \\ u_\varepsilon = u_\varepsilon^0(t=0) \end{cases} \tag{4.26}$$

with the appropriate boundary, solving this advection equation with the formerly described stabilized finite element method, the result of u_ε^t may not always maintain the metric property of the regularized hyperbolic tangent function, as the $||\nabla u_\varepsilon^t||_2$ computed may be different from $(1-(\frac{u_\varepsilon^t}{\varepsilon})^2)$. Inspired from the previous redistancing method, we introduce a new redistancing function to rebuild, the wanted u_ε^t. We introduce a fictitious time τ, and let us find $\alpha(\varepsilon, S, \tau)$, that has the same zero value as u_ε^t and is the result of

$$\begin{cases} \dfrac{\partial \alpha}{\partial \tau} + S(u_\varepsilon^t)\left[||\nabla \alpha||_2 - g(\alpha)\right] = 0 \\ g(\alpha) = 1 - (\dfrac{\alpha}{\varepsilon})^2 \\ \alpha^0(\varepsilon, S, \tau = 0) = u_\varepsilon^t(t) \text{ at } \tau = 0 \end{cases} \tag{4.27}$$

where $S(u_\varepsilon^t)$ is again the sign function, only dependent on $u_\varepsilon(t)$:

$$S(u_\varepsilon^t) = \begin{cases} 1 & \text{if } u_\varepsilon^t > 0 \\ 0 & \text{if } u_\varepsilon^t = 0 \\ -1 & \text{if } u_\varepsilon^t < 0 \end{cases} \tag{4.28}$$

As previously, we may rewrite Equation (4.27) as a pure advection equation:

$$\begin{cases} \dfrac{\partial \alpha}{\partial \tau} + \mathbf{U}\nabla\alpha = S(u_\varepsilon^t) \cdot g(\alpha) \\ \alpha^0(\varepsilon, S, \tau = 0) = u_\varepsilon^t \text{ at } \tau = 0 \end{cases} \tag{4.29}$$

being the velocity $\mathbf{U}$ given by:

$$\mathbf{U} = S(u_\varepsilon^t)\frac{\nabla\alpha}{||\nabla\alpha||_2} \tag{4.30}$$

The standard algorithm for classically advecting rebuilding interfaces is given below.

As suggested in [82, 83], a convective redistancing technique may be applied to simultaneously advect and reinitialize u_ε. For that, a parameter λ is introduced:

$$\lambda = \frac{\partial \tau}{\partial t} = \frac{\Delta\tau}{\Delta t} \tag{4.31}$$

while *the end of the computation has not been attained* **do**
  Solve the Navier-Stokes equation to obtain the velocity $\mathbf{v}$.
  Solve the advection equation (4.26) with u_ε^{t-1}, to obtain u_ε^t.
  Compute the sign function, $S(u_\varepsilon^t)$, as given by equation (4.28).
  Redistance by solving equation (4.27), with u_ε^t being the initial value of $\alpha^0 = u_\varepsilon^t$ to obtain α^τ, for a fixed number of iterations.
  Update the redistanced function, $u_\varepsilon^t = \alpha^\tau(\varepsilon, S, \tau)$.

This last will allow us to define a relationship between the real and the fictitious time step, since:

$$\frac{\partial \alpha}{\partial t} = \lambda \frac{\partial \alpha}{\partial \tau} \tag{4.32}$$

Taking Equations (4.26) and (4.27), and using this last expression, one may rewrite:

$$\begin{cases} \dfrac{\mathbf{d}u_\varepsilon}{\mathbf{d}t} + \lambda S(u_\varepsilon^t)\left[||\nabla u_\varepsilon||_2 - g(u_\varepsilon)\right] = 0 \\ u_\varepsilon^0 = u_\varepsilon(t=0) \text{ at } t = 0 \end{cases} \tag{4.33}$$

with:

$$\frac{\mathbf{d}u_\varepsilon}{\mathbf{d}t} = \frac{\partial u_\varepsilon}{\partial t} + \mathbf{v} \cdot \nabla u_\varepsilon \tag{4.34}$$

Finally, the convected redistancing equation used to compute interface displacement under a computed velocity field may be written as:

$$\begin{cases} \dfrac{\partial u_\varepsilon}{\partial t} + (\mathbf{v} + \lambda \mathbf{U}) \cdot \nabla u_\varepsilon = \lambda S(u_\varepsilon^t) g(u_\varepsilon) \\ u_\varepsilon^0 = u_\varepsilon(t=0) = \text{ at } t = 0 \end{cases} \tag{4.35}$$

This means that both the interface motion and the smooth and regularized hyperbolic tangent function are displaced and constructed at the same time, by solving Equation (4.35).

4.6 Numerical examples

Before coupling the Navier-Stokes multiphase solver with the immersed image and redistancing method, it is necessary to illustrate the type of performed computations, by simulating classical fluid-structure interaction and moving interfaces problem, which allow the validation of the incompressible Navier-Stokes solver and also present the accuracy and advantages of using anisotropic mesh adaptation to build simultaneously the mesh and the solution fields.

4.6.1 Flow around a cylinder

The first test is shown in Figure 4.3, for a computational rectangular domain of size $[32.5d, 10d]$, being d the cylinder of diameter, $d = 0.1$. The cylinder is located at $[12.5d, 5d]$, and is a rigid fixed body. The fluid flows crosses from the left side to the right, with the boundary conditions detailed in the Figure. Changing the value of the imposed velocity will modify the studied Reynolds number.

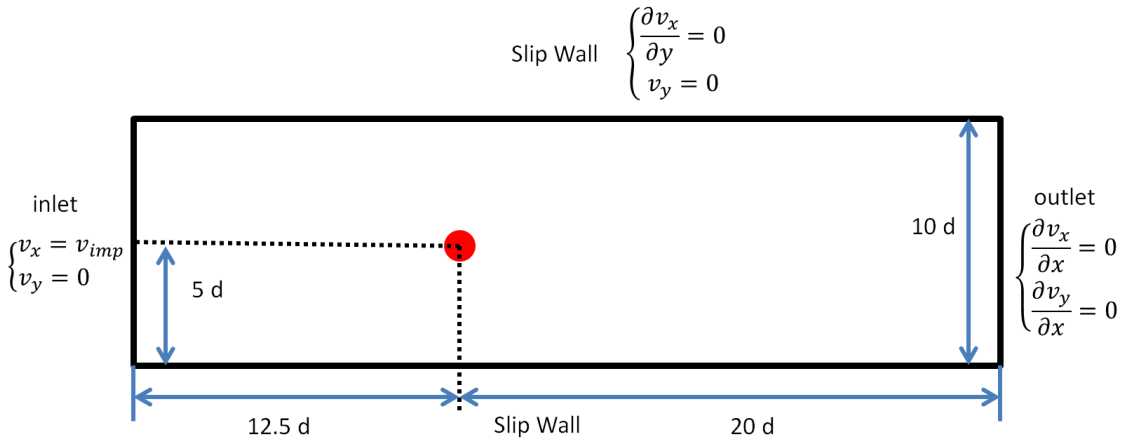


Figure 4.3: Size and geometry of the whole computational domain for the flow around a cylinder test.

Our modified level-set function, u_ε, with $\varepsilon = 0.0025d$, represents the cylinder through an implicit function allowing the distinction between the solid and fluid phase functions, as shown in Figure 4.4.

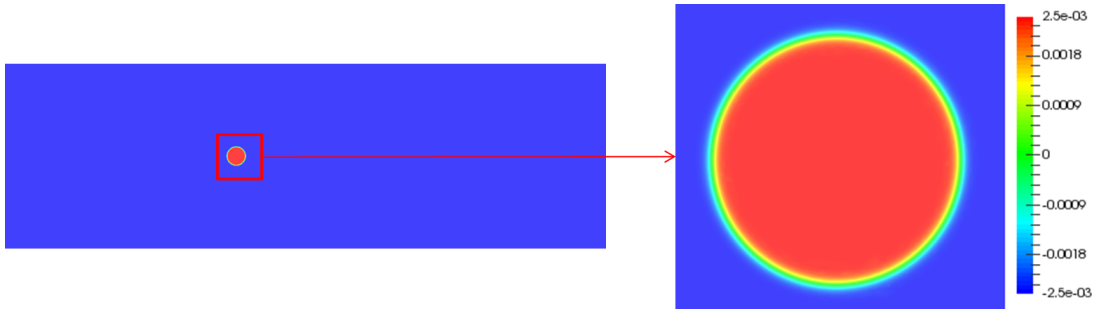


Figure 4.4: Modified level-set function used to represent the cylinder and fluid phases, and zoom of this function to better distinguish the solid-fluid transition representation in the mesh.

The solid phase function may then be used to obtain the physical parameters. The densities ρ_{fluid}, $\rho_{cylinder}$ and the dynamic viscosities η_{fluid}, $\eta_{cylinder}$ are used to define ρ and η in the whole computational domain with the mixture law (Equation (4.11)). The number of nodes in the mesh has been fixed at $N = 10000$ and the estimated error will be computed to adapt this mesh on both u_ε and on the computed velocity field, $\mathbf{v}$. The algorithm followed to solve the whole FSI problem is given below.

Input: Input the initial mesh $\mathcal{H}$, the geometry and the boundary conditions, the thickness ε, number of nodes N, the physical parameters of each phase, $\rho_{cylinder}, \rho_{flow}, \eta_{cylinder}, \eta_{flow}$, the initial velocity $\mathbf{v}^0$ and the time step Δt.

Output: The computed velocity field $\mathbf{v}$ and the adapted anisotropic mesh.

while $t <$*total computing time* **do**
- Compute u_ε^t on the current mesh.
- Compute the physical parameters distribution on the entire domain, ρ, η using a mixture law method.
- Solve the multiphase Navier-Stokes equations, to obtain $\mathbf{v}$, using a VMS finite element method.
- Construct the adaptive anisotropic mesh, by estimating the error on both u_ε^t and on $\mathbf{v}$.
- Update the new mesh, and the computed velocity field $\mathbf{v}$.
- $t = t + \Delta t$.

We have firstly considered Reynolds numbers in the permanent regimes ($5 \leq Re \leq 49$), where two stationary recirculations arise behind the cylinder. Then, subsequent computations concerned the periodic regime ($50 \leq Re \leq 190$). Even if still laminar, there are two periodic Von Kármán vortex formation.

For the permanent regime investigation, we have fixed the $Re = 10, 20, 30, 40.$ and have two computed parameters: the length of the recirculation zone, L_r, and the drag coefficient, C_D. L_r is defined as the downstream distance to the vortex shedding, where the velocity is null, measured on the central line. On the other hand, C_D is defined by Equation (4.36).

$$C_D = \frac{F_{Drag}}{1/2\rho v_{imp}^2 d} \tag{4.36}$$

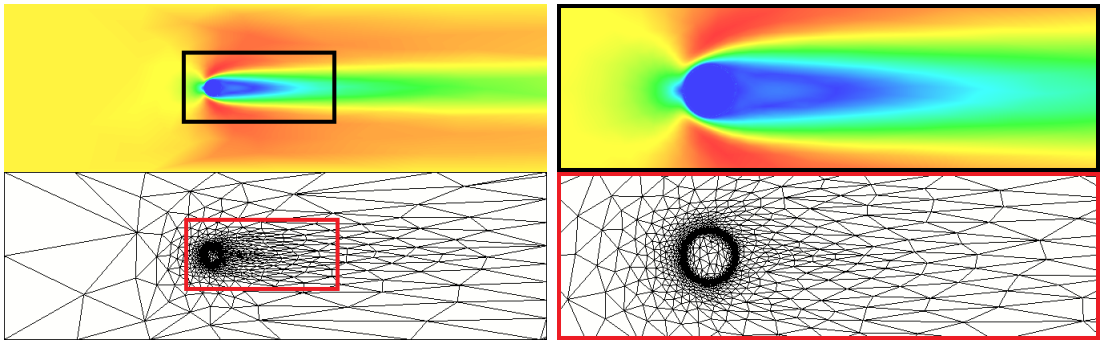
where v_∞ is the maximum inflow velocity, F_{Drag} is the x-component of $\mathbf{F}$, the total force on the structural surface S, and $\mathbf{S}$ is the deviatoric stress tensor:

$$\mathbf{F} = \int_S (-p\mathbf{I} + \mathbf{S}) \cdot \mathbf{n} ds \tag{4.37}$$

Figure 4.5 illustrates the stationary computed velocity field $\mathbf{v}$, for a condition without vortex formation, showing that mesh adaptation adapts well both on the velocity field $\mathbf{v}$ and on u_ε. Steady-state velocity is just as expected for permanent regimes.

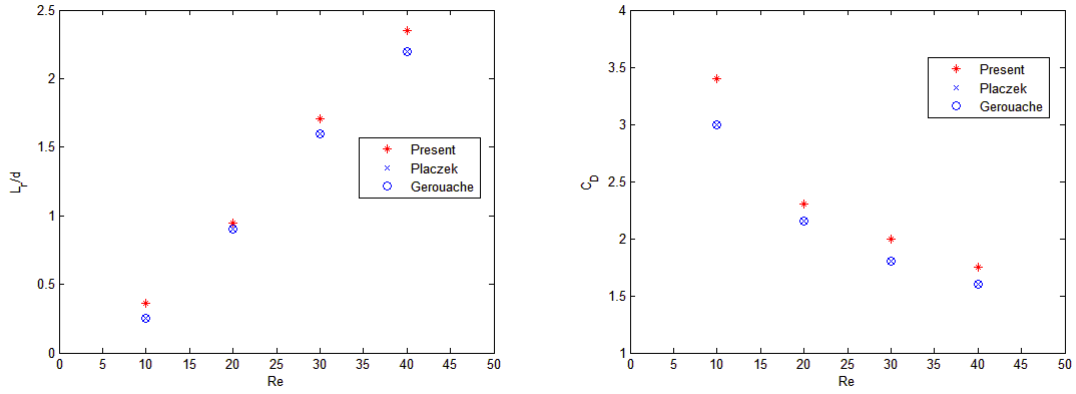
Parameters L_r and C_D were measured for $Re = 10, 20, 30, 40$, and are plotted in Figure 4.6. Their results were compared with ones from [84, 85], where our approach is very close to these references.

If the Reynolds number is increased to $Re = 60, 80, 100, 120$, theoretically two rows of vortexes periodically will shed from the cylinder. This periodic shedding may lead to oscillations on the aerodynamic parameters. In fact, it will reflect the computation of the drag coefficient, C_D, and of the lift coefficient, C_L. The lift coefficient C_L is given by



(a) Velocity field $\mathbf{v}$ and adapted mesh (b) Zoom around the cylinder

Figure 4.5: Illustration of the computed velocity field $\mathbf{v}$ and of the adapted anisotropic mesh at $Re = 40$.



(a) Adimentional recirculation length, L_r/d (b) Drag coefficient, C_D

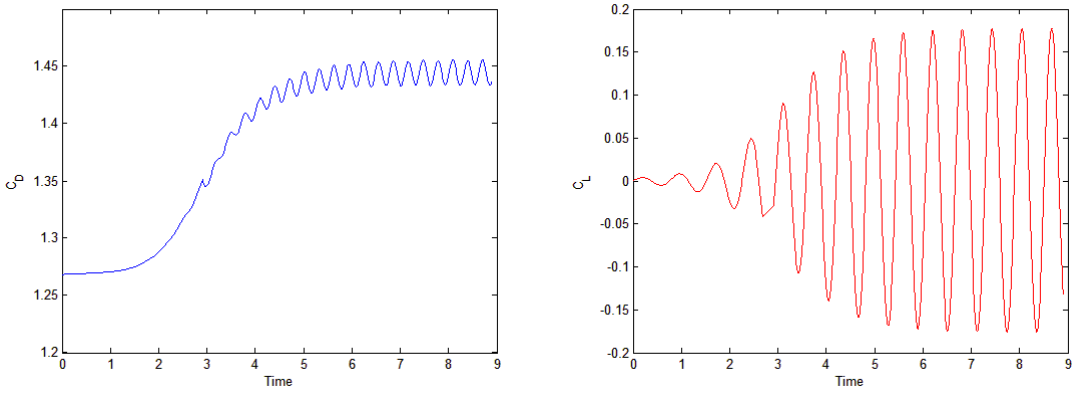
Figure 4.6: Comparison of the recirculation length L_r/d and of the drag coefficient C_D for the permanent regime ($5 \leq Re \leq 49$).

$$C_L = \frac{F_{Lift}}{1/2\rho v_{imp}^2 d} \tag{4.38}$$

where F_{Lift} is the y-component of $\mathbf{F}$. We have plotted C_D and C_L as a function of time in Figure 4.7 for $Re = 120$. We observe that the frequency of the drag coefficient oscillations is twice the one of the oscillation of the lift coefficient. To analyze this, one other parameter was introduced, the Strouhal number, S_t, which is defined as:

$$S_t = f_s \frac{d}{v_x} \tag{4.39}$$

where the Strouhal frequency f_s is the lift coefficient frequency. Figure 4.8 illustrates both the obtained velocity field $\mathbf{v}$ and the adapted anisotropic mesh over one Strouhal period at $Re = 120$. In this case, we observe the periodic vortex shedding formulation behind the cylinder, and the adapted anisotropic mesh enriches with more nodes this high velocity gradient area. To validate these periodic



(a) Coefficient C_D (b) Coefficient C_L

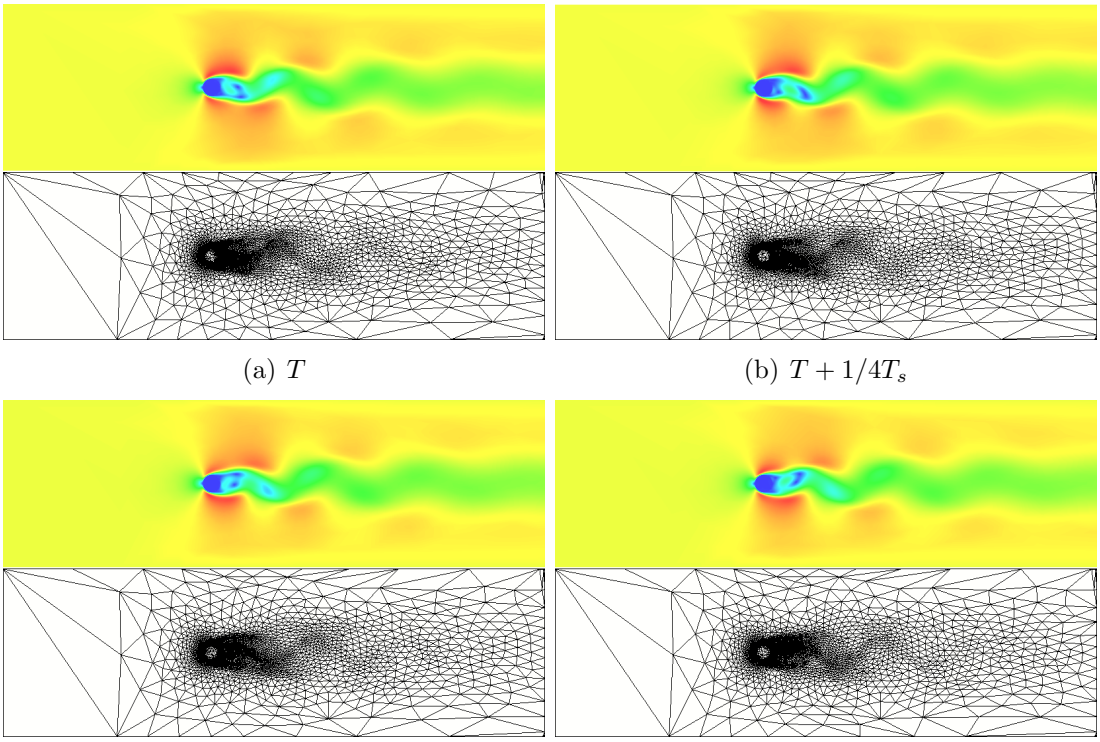
Figure 4.7: Aerodynamic coefficients C_D and C_L at $Re = 120$.

investigation results, we use the maximum value of C_L, the $C_{L,max}$, since the average of this parameter is approximate null, but also the Strouhal number, S_t, at $Re = 60, 80, 100, 120$. Figure 4.9 illustrates the comparison of our results with other literature's results [84] and with analytical ones [86].

These comparisons show that $C_{L,max}$ results are below the analytical ones, but closer to them than other state of the art. To check on the reason of this difference, one should do a reversibility analysis on the thickness ε and on the mesh number of nodes to check what would be the best representation. This work has been done with the tools developed in [71]. The Strouhal number, on the other hand, is closer to the analytical one.

This approach can handle 2D tests, but also 3D situations. Let us extend the 2D geometry to a 3D volume, with a width size of $5d$. Figures 4.10 and 4.11 show the computed 3D velocity field and the adapted anisotropic mesh, for $N = 100000$ and at the Reynolds number $Re = 40$ and 100.

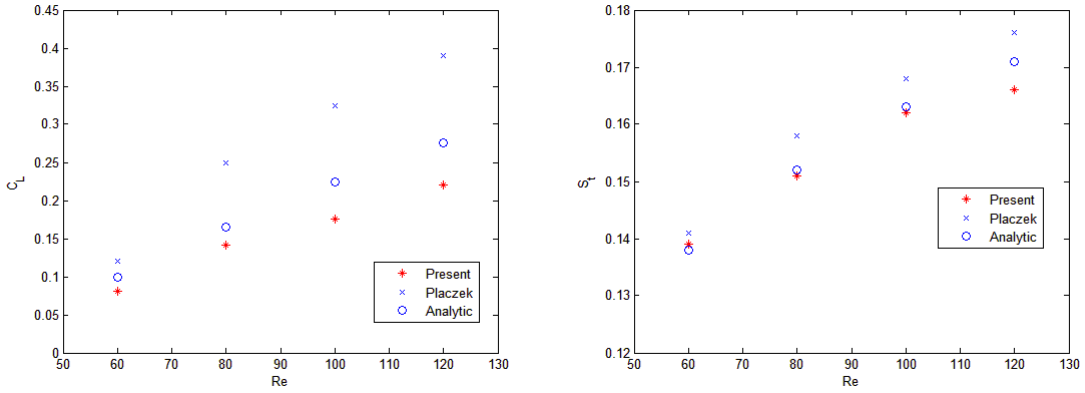
Obviously, the velocity attains the steady-state for this permanent regime, at $Re = 40$, and vortex shedding is observed for the periodic regime, at $Re = 100$. One observes also how the 3D mesh adapts anisotropically, both on the velocity and on the hyperbolic tangent function fields.



(a) T (b) $T + 1/4T_s$

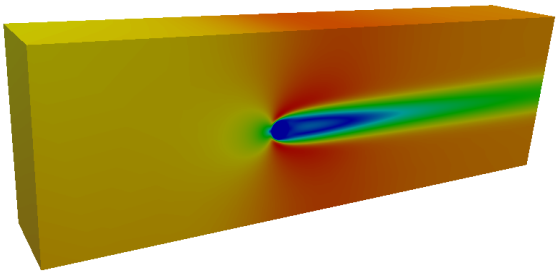
(c) $T + 2/4T_s$ (d) $T + 3/4T_s$

Figure 4.8: Illustration of the computed velocity field $\mathbf{v}$ and of the corresponding adapted anisotropic mesh at $Re = 120$ over one Strouhal period.

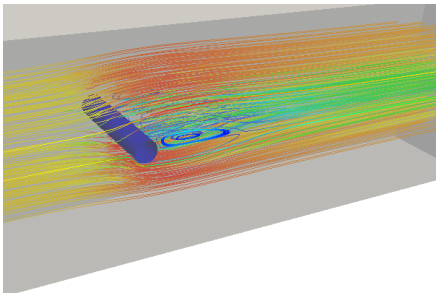


(a) Maximal lift coefficient, $C_{L,max}$ (b) Strouhal number of S_t

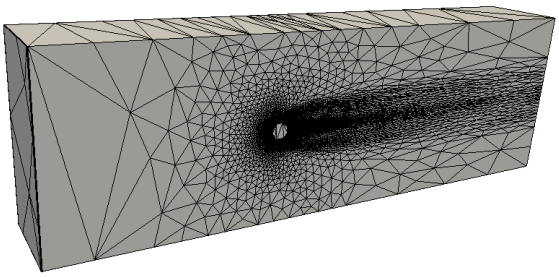
Figure 4.9: Comparison of the maximum lift coefficient, $C_{L,max}$, and of the Strouhal number, S_t, for the periodic regime ($50 \leq Re \leq 190$).



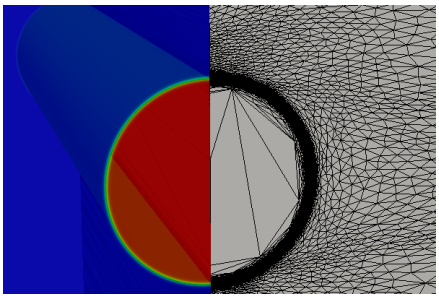
(a) Computed velocity field, $\mathbf{v}$



(b) Streamlines of the velocity past the cylinder

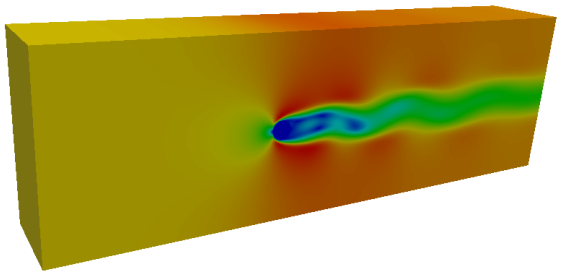


(c) Adapted anisotropic mesh

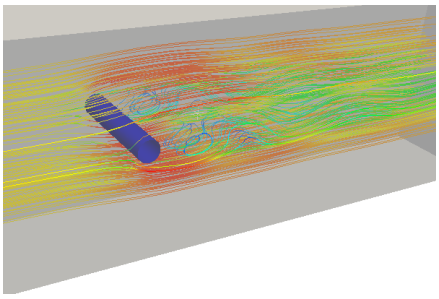


(d) u_ε

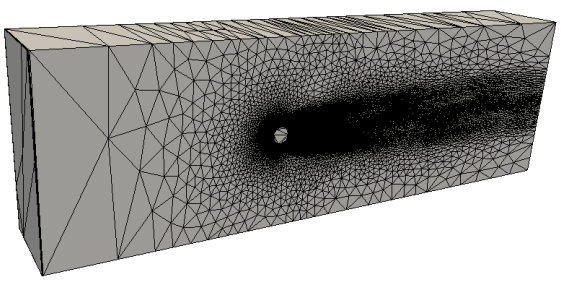
Figure 4.10: Illustration of the computed velocity field $\mathbf{v}$, u_ε, as well as the adapted anisotropic mesh for $Re = 40$.



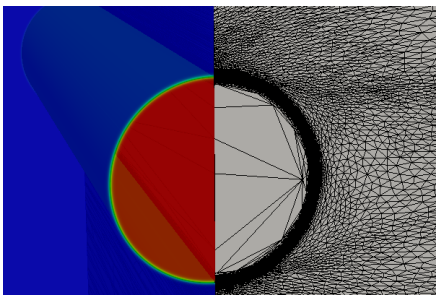
(a) Computed velocity field, $\mathbf{v}$



(b) Streamlines of velocity past the cylinder



(c) Adapted anisotropic meshes



(d) u_ε

Figure 4.11: Illustration of the computed velocity field $\mathbf{v}$, u_ε, as well as the adapted anisotropic mesh for $Re = 100$.

4.6.2 Fluid buckling

Let us now illustrate a case where there are also different phases, but that are not spatially fixed. Two-fluid flow is thus considered, by simulating the jet of a viscous fluid impacting a rigid plate. The geometry is given in Figure 4.12, with a rectangular injector located at a $height = 10d$, in a box of size $[10d \times 12d]$, where $d = 0.1m$.

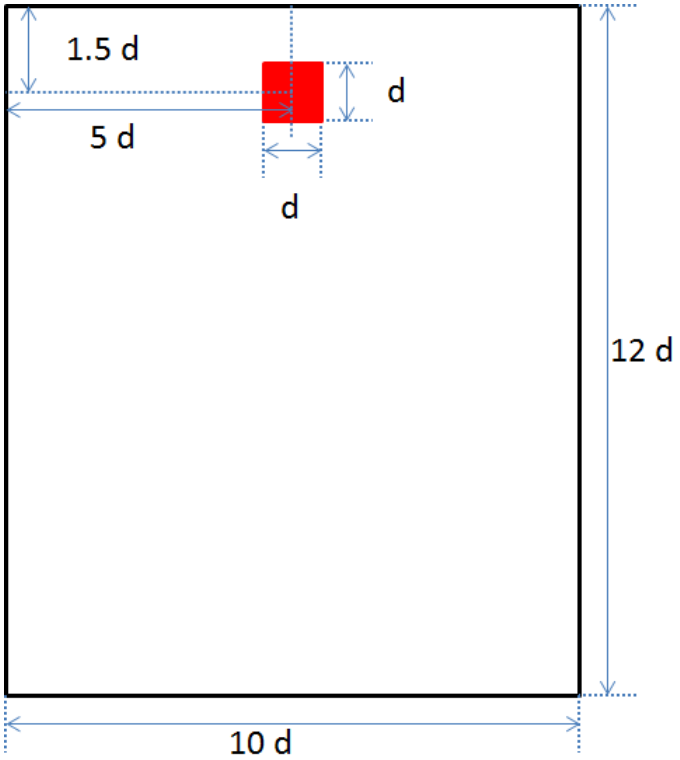


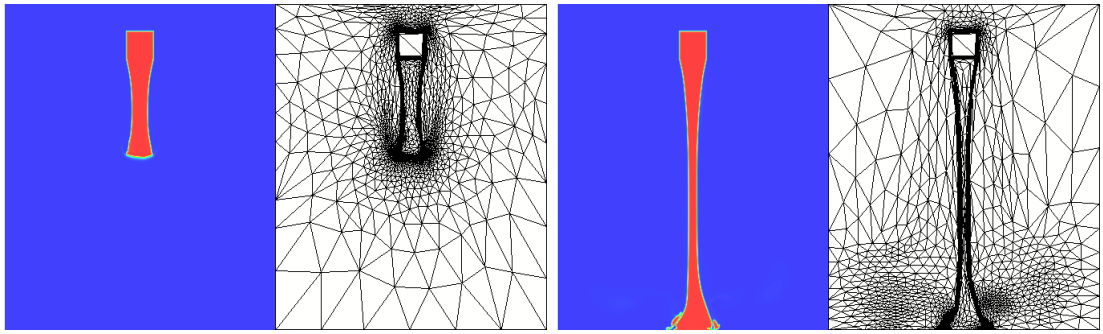
Figure 4.12: Geometry used for fluid buckling simulation, where $d = 0.1m$.

In this case, the gravity effect is an important factor and we have considered $g = 9.18m/s^2$. The moving convected level-set method has been used to compute u_ε^t and capture the moving fluid. In our simulation, the thickness was supposed $\varepsilon = 0.02d$ and represents the separation between fluid and air domains and is obtained by solving the convected redistancing equation (4.35). A small thickness ε may allow a very good representation of the configuration of the moving fluid. Parameters were defined as $\rho_{fluid} = 1800kg/m^3$ and $\eta_{fluid} = 500Pa \cdot s$, and the injection velocity is $\mathbf{v}_{imp} = (0, 0.25m/s)$, with the time step $\Delta t = 0.05$. Figure 4.13 shows the fluid buckling and the adapted anisotropic mesh throughout time.

We observe clearly the configuration of the moving liquid part, and its oscillations after an initial stable jet, captured thanks to the anisotropic adaptation.

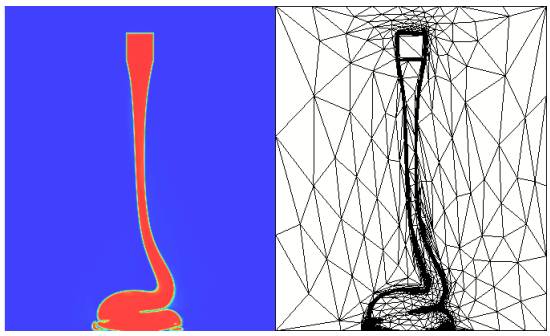
This application can also be extended to the 3D case, the rectangular injector becoming a cylinder one. In 3D, more nodes have been required, with N fixed at 100000, for a higher computational cost, in a logical way. Final illustrations of this last case are presented in Figure 4.14.

Detail of the validation on fluid buckling simulations using the followed methodology and performed by other co-workers has been well presented in [82, 83] and

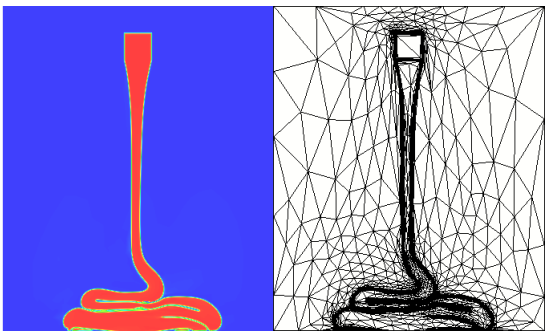


(a) $t = 10s$ (200 increments)

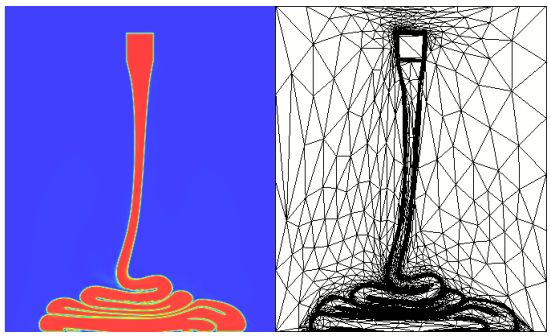
(b) $t = 20s$ (400 increments)



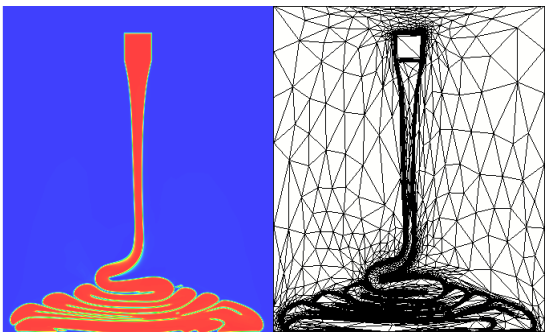
(c) $t = 30s$ (600 increments)



(d) $t = 40s$ (800 increments)



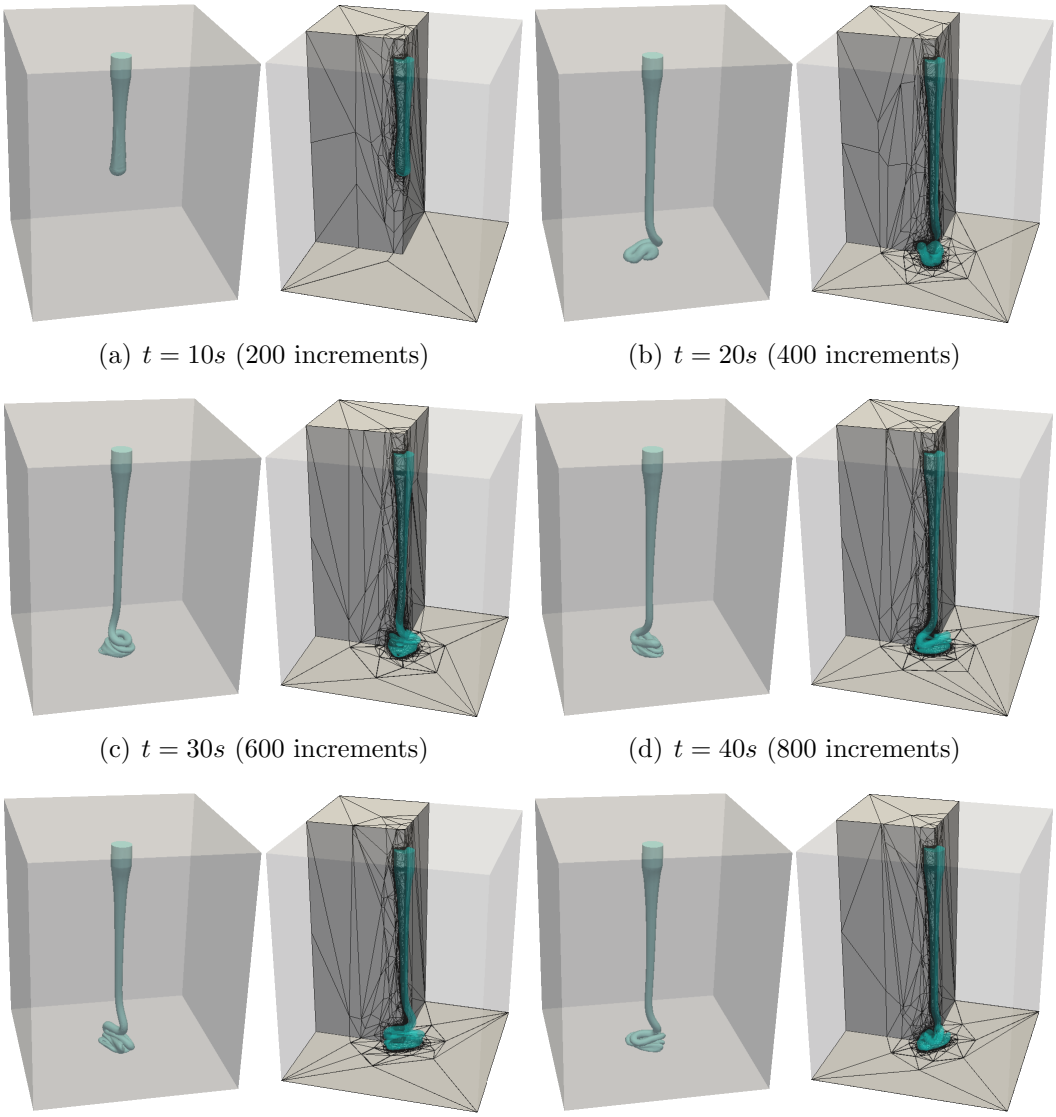
(e) $t = 50s$ (1000 increments)



(f) $t = 60s$ (1200 increments)

Figure 4.13: Illustration of 2D fluid buckling simulation results, as a function of time, with 10000 fixed nodes for the anisotropic adapted mesh.

were here given as simple illustrations of the presented numerical techniques.



(a) $t = 10s$ (200 increments) (b) $t = 20s$ (400 increments)

(c) $t = 30s$ (600 increments) (d) $t = 40s$ (800 increments)

(e) $t = 50s$ (1000 increments) (f) $t = 60s$ (1200 increments)

Figure 4.14: Illustration of 3D fluid buckling simulation, for a 100000 nodes anisotropic mesh.

4.7 Numerical simulations based on real images

In this section, we will illustrate the usage of the developed numerical methods through numerical simulations based on real images, which involve image processing, interpolation of the image on mesh, redistancing a modified level-set function, mesh adaptation and a multiphase Navier-Stokes solver. The image processing provides the geometry of the object of interest, interpolation of the image on the mesh links the image information and the discretization of the studied phases on the mesh. The redistancing method coupled to anisotropic adaptation allows an accurate description of these phases and provides also regularized level-set functions coupled to the mesh as input to the flow solver. Simulation results concern not only flow but include the fact that the three features (flow, geometry, mesh) are built simultaneously from the image and in an adapted way.

4.7.1 2D picture based simulations

In the industrial design domain, the first challenge is the numerical description of a given object, the accuracy being the key for the performance. A second challenge might be to be able to represent this object in the mesh format. To pursue both objectives, too much time may be spent. CAD software is used to create numerically the objects, followed by specific meshing tools to generate an adapted mesh. The methodology proposed in this work may help to easily achieve this goal at a minimum effort. For that, let us consider the example of a Formula 1 racing car, illustrated in Figure 4.15.

Figure 4.15: Original Formula 1.

We intend to simulate the air flow at high velocity, surrounding the Formula 1 vehicle. The first step is to segment the racing car from the color image. The original image has been convected with the three channels HSL (Hue, Saturation, Lightness), which are presented in Figures 4.16(a), (c) and (e). From these three channels, we have chosen specific thresholding values. For example, thresholding values$= (25, 30, 9)$ for (Hue, Saturation, Lightness) channels and have segmented different parts from them. We assembled these segmented images to obtain the overall vehicle description, Figure 4.16(g), which is very close to the description of

the racing car. To fill the empty domain and filter some noise, we have done Opening and Closing operations, the final image being given in Figure 4.16(h), which well describes the boundary of the racing car. Hence, this image data provides the segmentation of the car in the original image, which will give us the sign function for the redistancing procedure.

Let us now consider this final segmented racing car image, the white color representing the racing car and the ground, $\widehat{u}_{seg}$, shown in Figure 4.17, being both considered rigid solids. The black color is thus the region where air flows, from the left to the right side with the illustrated imposed boundary conditions. The image is of size (2000×500).

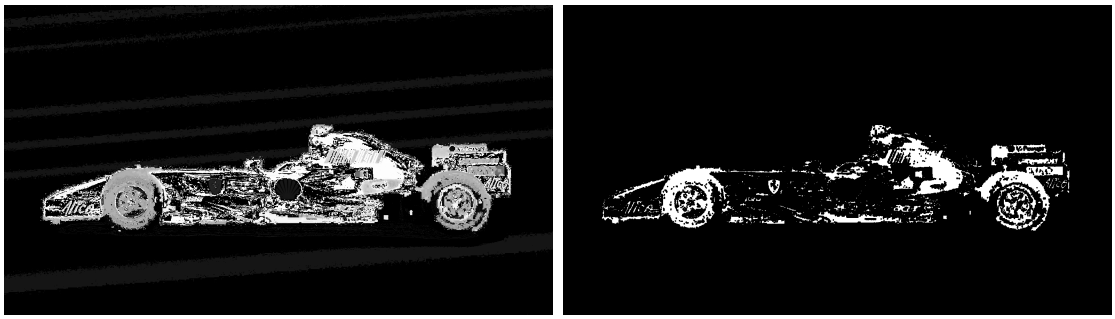
Parameters for the redistancing resolution, the Navier-Stokes solver and the mesh adaptation procedure are the following: the thickness ε and the fictitious time step are, respectively, $\varepsilon = 0.01, \Delta\tau = 0.00025$; the dynamic viscosity and the density of the solid and the fluid are $\mathbf{v}$, ρ_{solid} η_{solid} and ρ_{flow} η_{flow}; the time step is $\Delta t = 0.005$ for the Navier-Stokes solver; the number of nodes is fixed at $N = 10000$ for anisotropic mesh adaptation. The whole algorithm for this application is detailed below as follows.

Input: Original color image $\widehat{u}$, initial mesh $\mathcal{H}$ in dimension $[0,4] \times [0,1]$, $(\varepsilon = 0.01, \Delta\tau = 0.00025)$, inlet velocity $\mathbf{v}$, physical parameters of each phase, ρ_{solid} η_{solid} and ρ_{flow} η_{flow}, time step $\Delta t = 0.005$, $N = 10000$.

Output: The computed velocity field $\mathbf{v}$, the redistanced function u_ε^τ, and adapted anisotropic mesh.

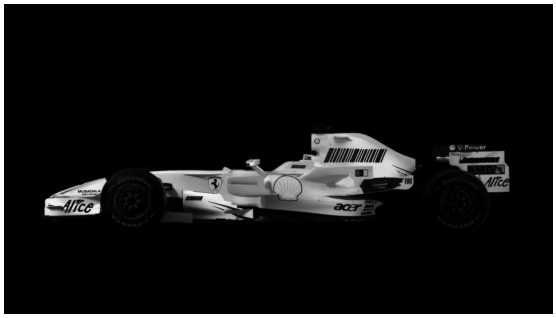
1 Image processing of the original color image $\widehat{u}$, to obtain the binary based segmented image $\widehat{u_{seg}}$.
2 Interpolate the segmented image $\widehat{u_{seg}}$ on the initial mesh, obtain the discretization function as the initial function $u_\varepsilon^0 = u_h$.
3 **while** $t <$ *total computing time* **do**
4 Compute the sign function $S(\widehat{u_{seg}})$ from the segmented image $\widehat{u_{seg}}$ on currant mesh.
5 Solve the redistancing equation, to obtain u_ε^τ.
6 Correct the sign of computed redistanced function u_ε^τ.
7 Compute the physical parametesr ρ η of the whole computational domain using a mixture law , by coupling the parameters of solid and fluid and the redistanced function u_ε^τ.
8 Solve the multiphase Navier-Stokes equations, to obtain the computed velocity field $\mathbf{v}$ of the flow, using a VMS finite element method
9 Construct the adaptive anisotropic mesh $\widetilde{\mathcal{H}}$, to adapt the convected level-set function u_ε^τ and the computed velocity field $\mathbf{v}$.
10 Update the new mesh, redistanced function u_ε^τ and computed velocity field $\mathbf{v}$.
11 $t = t + \Delta t$,

At each increment, the redistancing procedure, the multiphase Navier-Stokes solver, and mesh adaptation are launched. After 60 increments, at $t = 0.3$, the re-

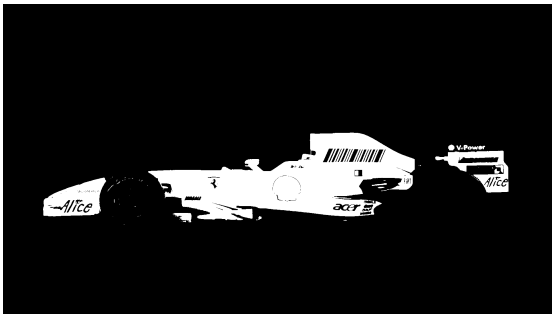


(a) H channel

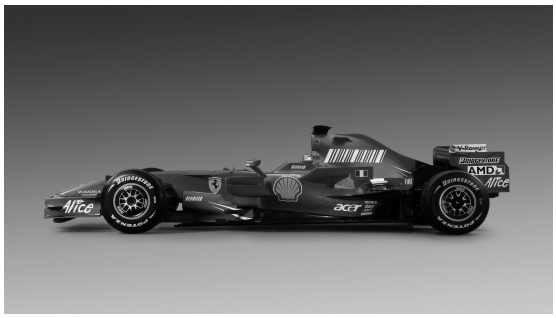
(b) Segmentation for the H channel



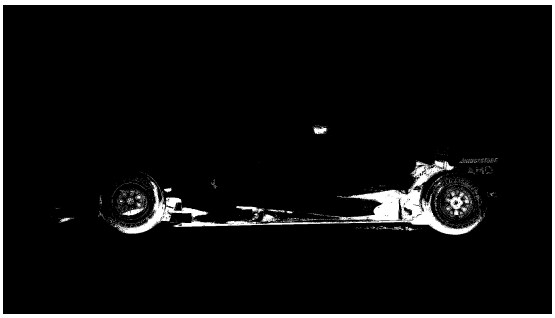
(c) S channel



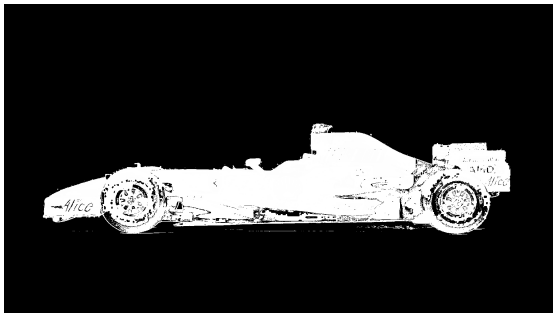
(d) Segmentation for the S channel



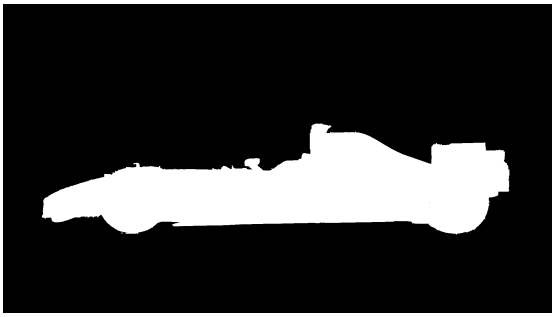
(e) L channel



(f) Segmentation for the L channel



(g) Assembling of segmented images



(h) Final segmented image

Figure 4.16: HSL channels built from the original image and their segmentation, as well as the final segmented image.

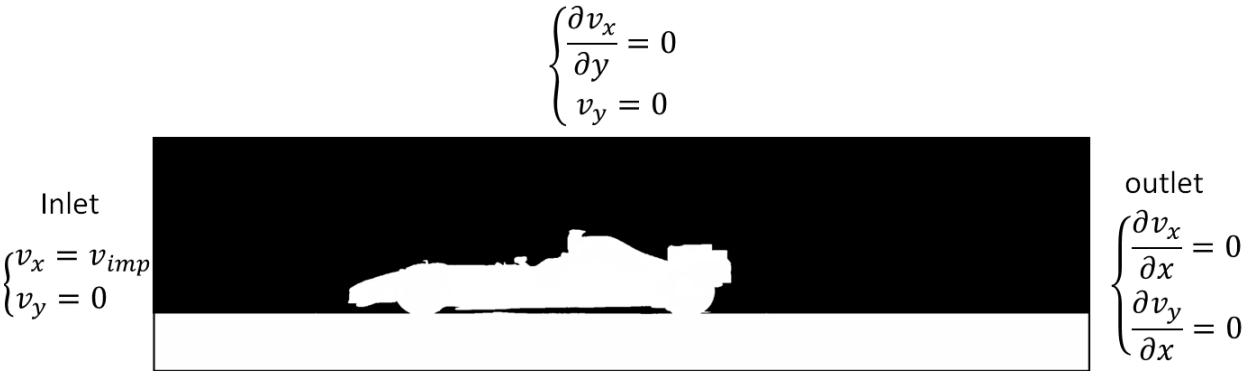
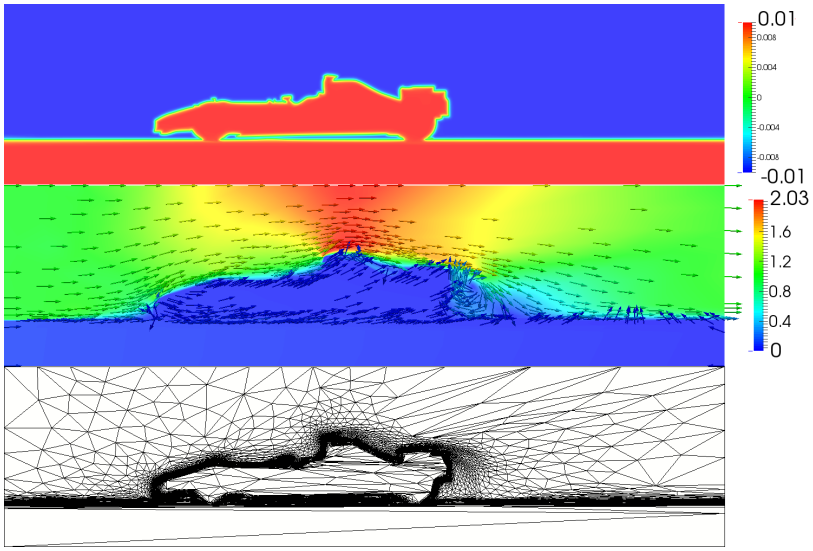


Figure 4.17: Segmented racing car image, and implemented boundary conditions.

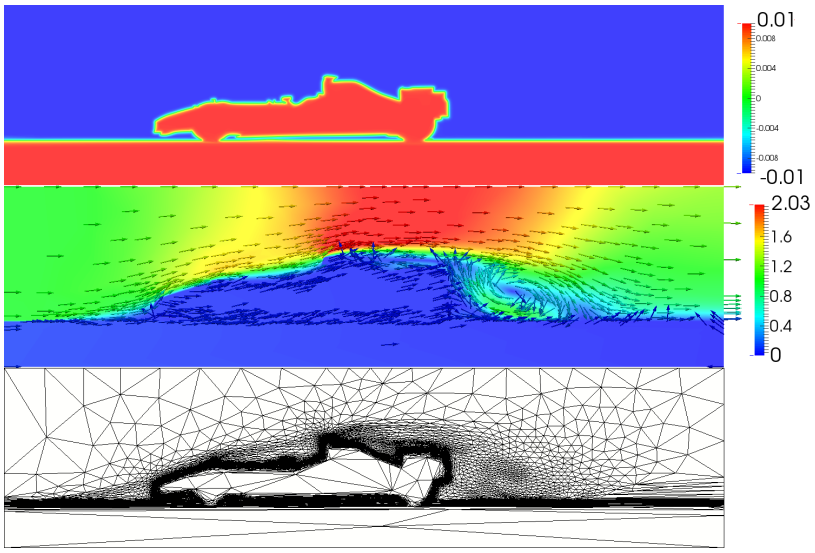
distancing function u_ε^τ has converged. Parameters of the whole computation domain are stable, and the Navier-Stokes solver provides a velocity field that may still vary due to turbulence and unsteady effects, accompanied by changes in the mesh. At $Re = 300$ and $t = 2.5s$ (500 increments), the redistanced level-set function u_ε^τ, the computed velocity field $\mathbf{v}$ and the adapted anisotropic mesh (with $N = 10000$) are illustrated in Figure 4.18. This computation run on 3 cores during 25 minutes.

We observe that the redistanced function well describes the boundary of the solid part, generating an accurate distribution of the physical properties, generated through the mixture law and used in the multiphase Navier-Stokes solver. The mesh at the different increments, well converge stowards an adapted solution, both on u_ε^τ, but also on $\mathbf{v}^t$.

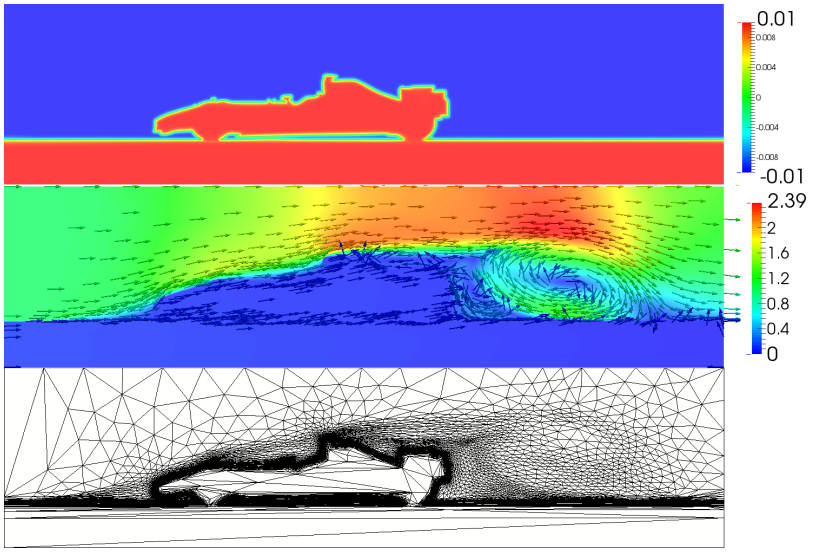
After this 2D illustration, which shows a direct application of our image to mesh tools, a 3D case concerning a real sample will be detailed.



(a) $t = 0.5$ (100 increments)



(b) $t = 1.5$ (300 increments)



(c) $t = 2.5$ (500 increments)

Figure 4.18: Redistanced function, computed velocity field and adapted anisotropic mesh, for $Re = 300$ at $t = 0.5, 1.5, 2.5s$

4.7.2 3D image based simulation

Let us introduce the 3D simulation of flow on an urban environment, where the geometry is given by images issued from *GIS* (Geographic Information Systems) data bases. These type of computation is often used for studies of urban climate, air pollution or city planning. A very important and difficult issue concerns the reconstruction of a 3D numerical sample that well contains all the geographic characteristics. One solution is to build the 3D image, like an "urban tomography". For that, we use the 2D image of the urban landscape and the height values corresponding to each pixel (that is the data contained in the image) to generate the 3rd dimension and the voxel value, as explained hereafter.

Figure 4.19, $\widehat{u}$, shows a 2D image of the Nantes urban description, obtained through *BDTopo* and IRSTV, Institut des Sciences et Techniques de la Ville, where the pixel grey-scale represents the distribution of the building's height. Hence, the relation between the pixel values and the height of the buildings is given by the following equation:

$$\text{height} = -0.2257 \times \widehat{u}(Pixel) + 57.228 \tag{4.40}$$

Figure 4.19: 2D image representing the urban (building) description of Nantes, obtained using the *GIS*'s database of *BDTopo*, through IRSTV.

To illustrate the followed methodology, we have chosen a part of this large image,

named again, $\widehat{u}_{2d}$, represented by a red rectangle in Figure 4.19, where Ecole Centrale de Nantes is located, for which a photo is shown in 4.20(a). This sub-image is also drawn in Figure 4.20(b), and has as dimensions (528×392) pixels. In reality, it represents a $528 \times 392\text{m}^2$ real surface.

To perform a 3D simulation, we have first created a 3D image of dimensions $(528 \times 392 \times 50)$, by extending the 2D based image. For that, each building's third dimension is completed to its own height, given by Equation (4.40). The rest of the voxels on this 3D image, belonging to the air, are given the 0 value. Finally, this allows plotting the 3D buildings and a 3D new image, $\widehat{u}$, as shown in Figure 4.20(c).

After the image construction procedure has been done, we consider the building parts as rigid bodies, in white color, the rest as the fluid flowing in black. The voxel values $\widehat{u}(Voxel)$ are defined as follows:

$$\begin{cases} \widehat{u}(Voxel) = 255 & \text{for the buildings} \\ \widehat{u}(Voxel) = 0 & \text{for the surrounding environment} \end{cases} \tag{4.41}$$

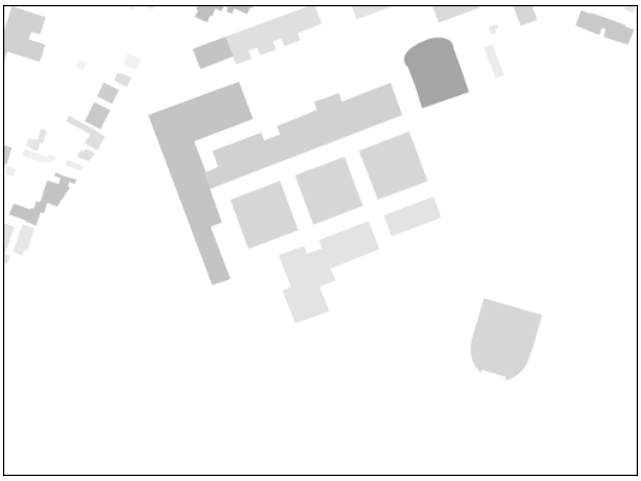
As previously presented, Morph-M directly provides us the distance function as an image, $\widehat{u_d}$. Then, we have interpolated it on a mesh of dimensions $[0, 5.27] \times [0, 3.91] \times [0, 0.49]$ to obtain: $u_d(X^i) = \widehat{u_d}(Voxel^k)/100$. Finally, the initial hyperbolic tangent function, $u_\varepsilon^0 = u_\varepsilon(u_d, \varepsilon)$ has been computed in the mesh as follows:

$$u_\varepsilon = \varepsilon \tanh(\frac{u_d}{\varepsilon}) \tag{4.42}$$

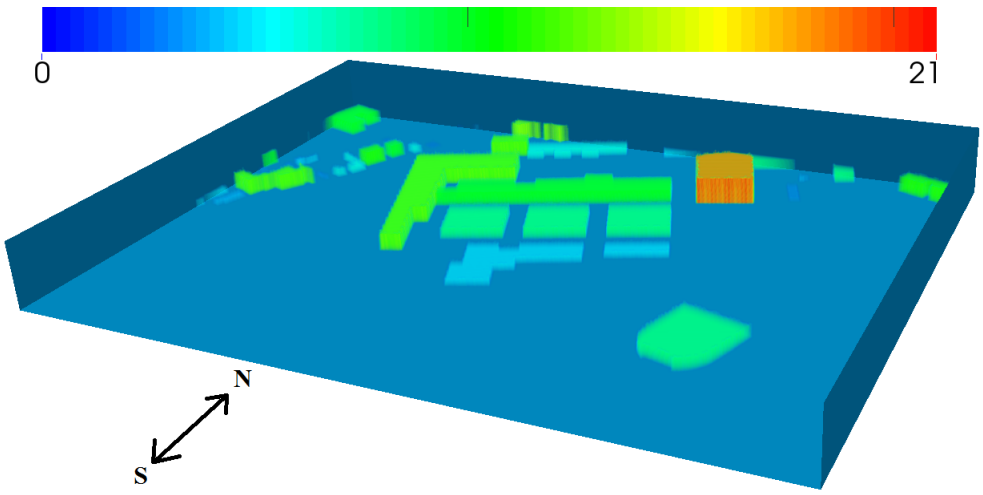
Figure 4.21(a) shows the image of the signed distance function to the buildings and air interface, u_d, obtained using Morph-M. Below, Figure 4.21(b), we have represented the computed hyperbolic tangent function, $u_\varepsilon(u_d, \varepsilon)$, after mesh interpolation.

Since the initial redistanced phase function is very close to the final solution, most of the increments are spent for the multiphase Navier-Stokes equation resolution and for mesh adaptation. For the redistancing scheme, we have input ($\varepsilon = 0.01, \Delta\tau = 0.00005$) and an imposed number of nodes N of 400000. We have performed 25 redistancing-adaptations before solving the flow, to well capture only the geometry. Following the previously introduced algorithm steps, we have launched the flow solver increments, with a time step $\Delta t = 0.01$. At $T = 2.5s$ (250 increments), the streamlines of the velocity field are illustrated in Figure 4.22, showing complex flow patterns, like the vortexes around the buildings. This type of information may be used to improve the design of buildings and city planning, even if here it remains only an illustration of an image-mesh-simulation methodology.

(a) Photo of the studied area

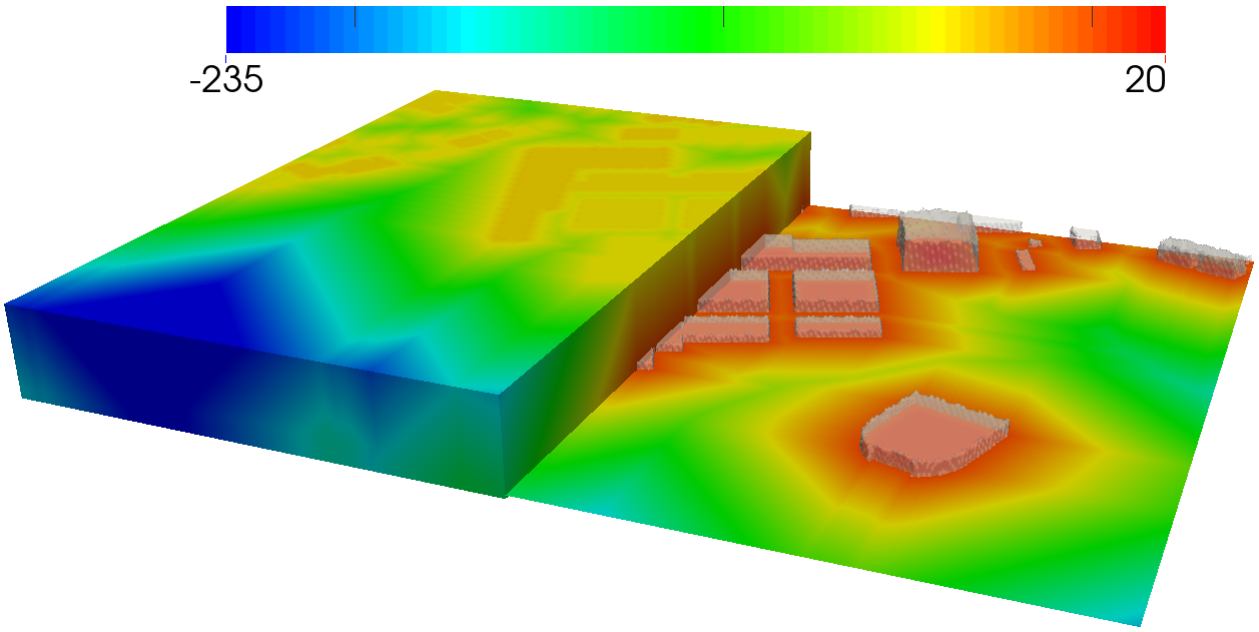


(b) 2D *BDTopo* image

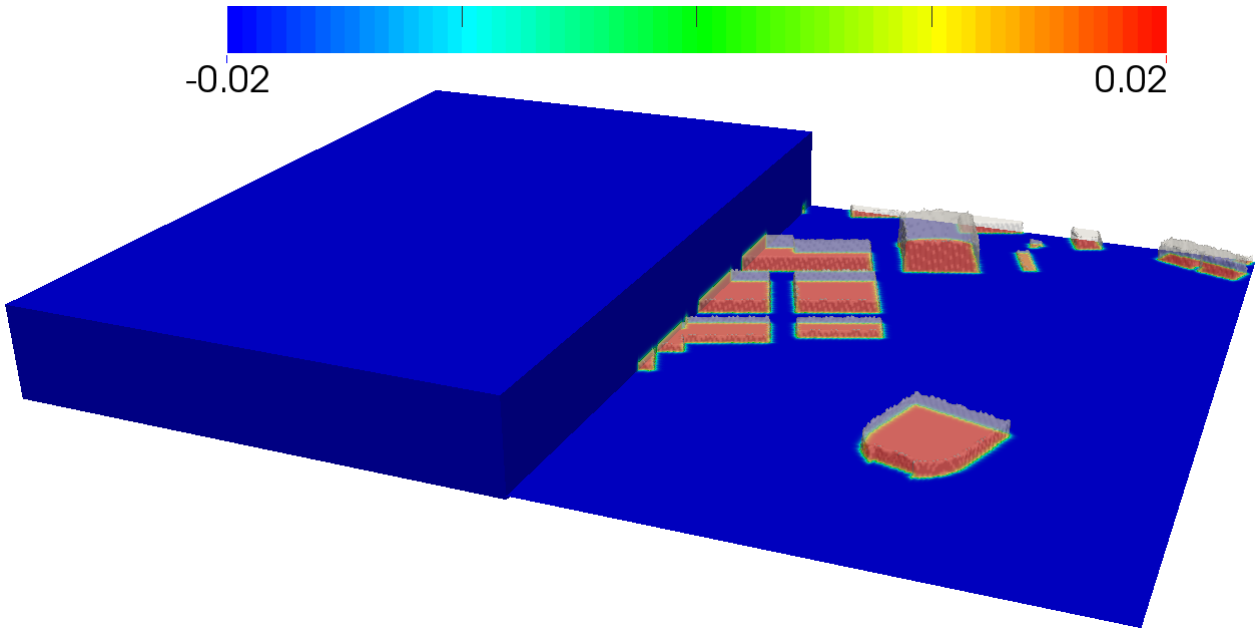


(c) 3D reconstruction of the studied area, the scale representing the height

Figure 4.20: 3D image created by extending a 2D *GIS* view.

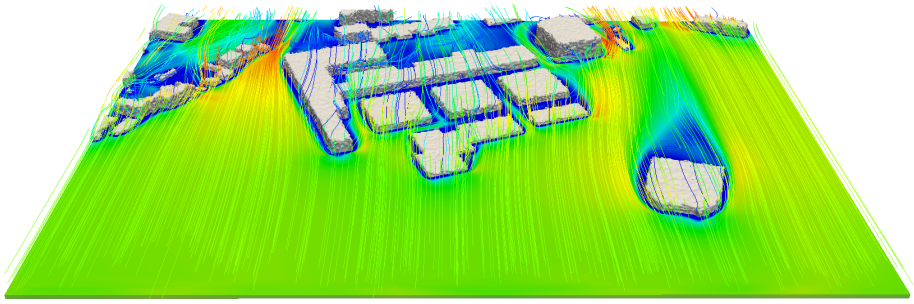


(a) Image of the distance field, $\widehat{u_d}$

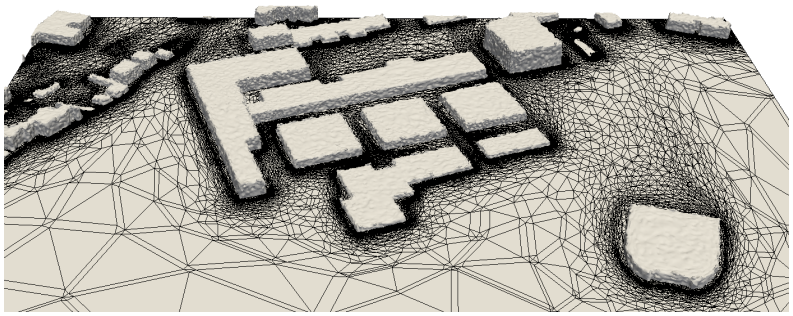


(b) Hyperbolic tangent function, computed on the mesh $u_\varepsilon(u_d, \varepsilon)$

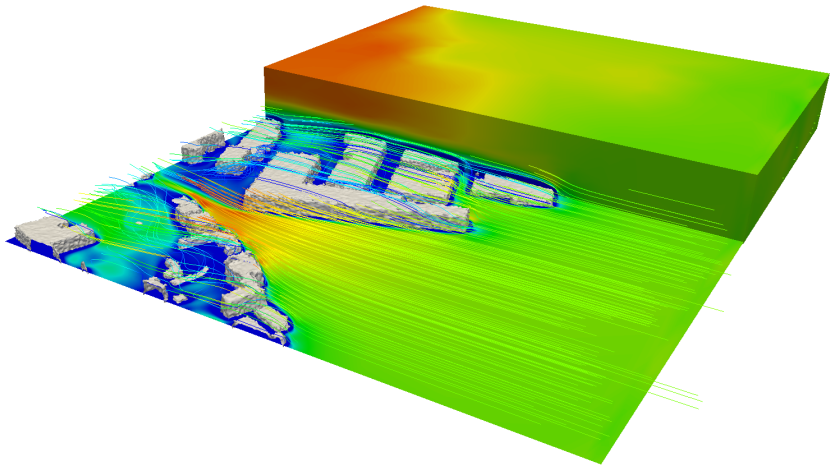
Figure 4.21: Image of the signed distance function, $\widehat{u_d}$, and computed hyperbolic tangent function, interpolated on the mesh, u_ε.



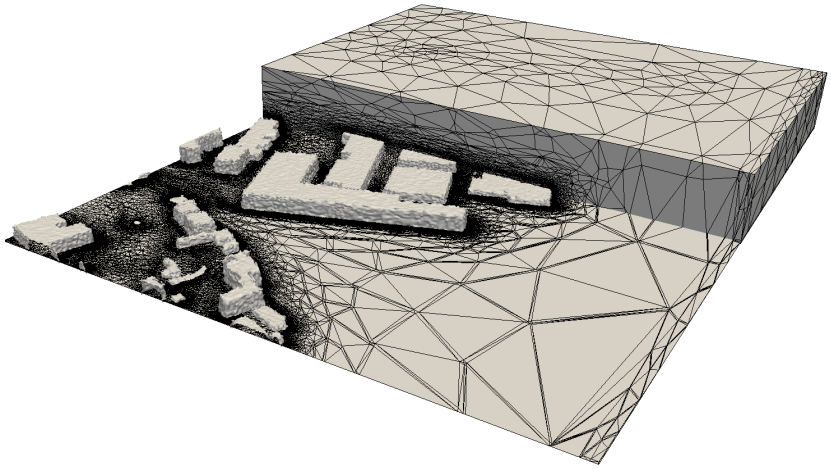
(a) Computed velocity field and flow streamlines



(b) Adapted anisotropic mesh



(c) Computed velocity field and flow streamlines



(d) Adapted anisotropic mesh

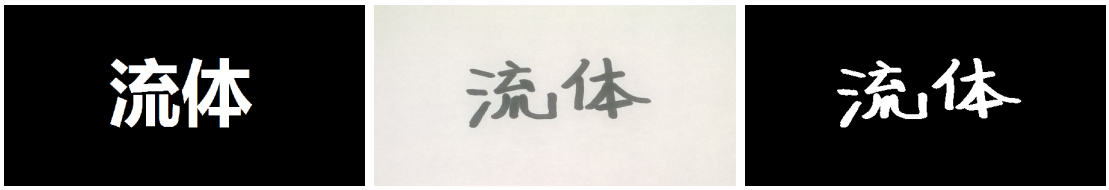
Figure 4.22: Air flow on an urban environment, with a representation of the computed velocity field and flow streamlines around the buildings represented in an implicit way and the adapted anisotropic mesh.

4.7.3 2D flow simulations on Paint and Phone images

In this section, we introduce numerical simulations based on images, obtained in a different way than the ones presented previously. Firstly, we have drawn an image using Microsoft Paint, in an accurate and controllable way, with pixel values that are fixed and easily define the different objects. Secondly, an image has been hand-made on paper and has been photographed after using a SmartPhone, to produce the used picture. The advantage of this method is that it is easily done, but a good quality of the image is not guaranteed, often needing image treatment and processing.

4.7.3.1 Fluid-structure interaction flow simulation

Let us consider two images representing the same type of object: the Chinese writing of the word "flow". The first image has been drawn with Microsoft Paint and is represented in Figure 4.23(a), where the white color represents the rigid body. Due to the good accuracy on the representation, this image does not need to be further processed. The second image has been drawn on paper, and then photographed with a SmartPhone, as illustrated in Figure 4.23(b). This image has very good quality thanks to the camera of the SmartPhone and has been processed initially by a thresholding technique. The final used image is shown in Figure 4.23(c).

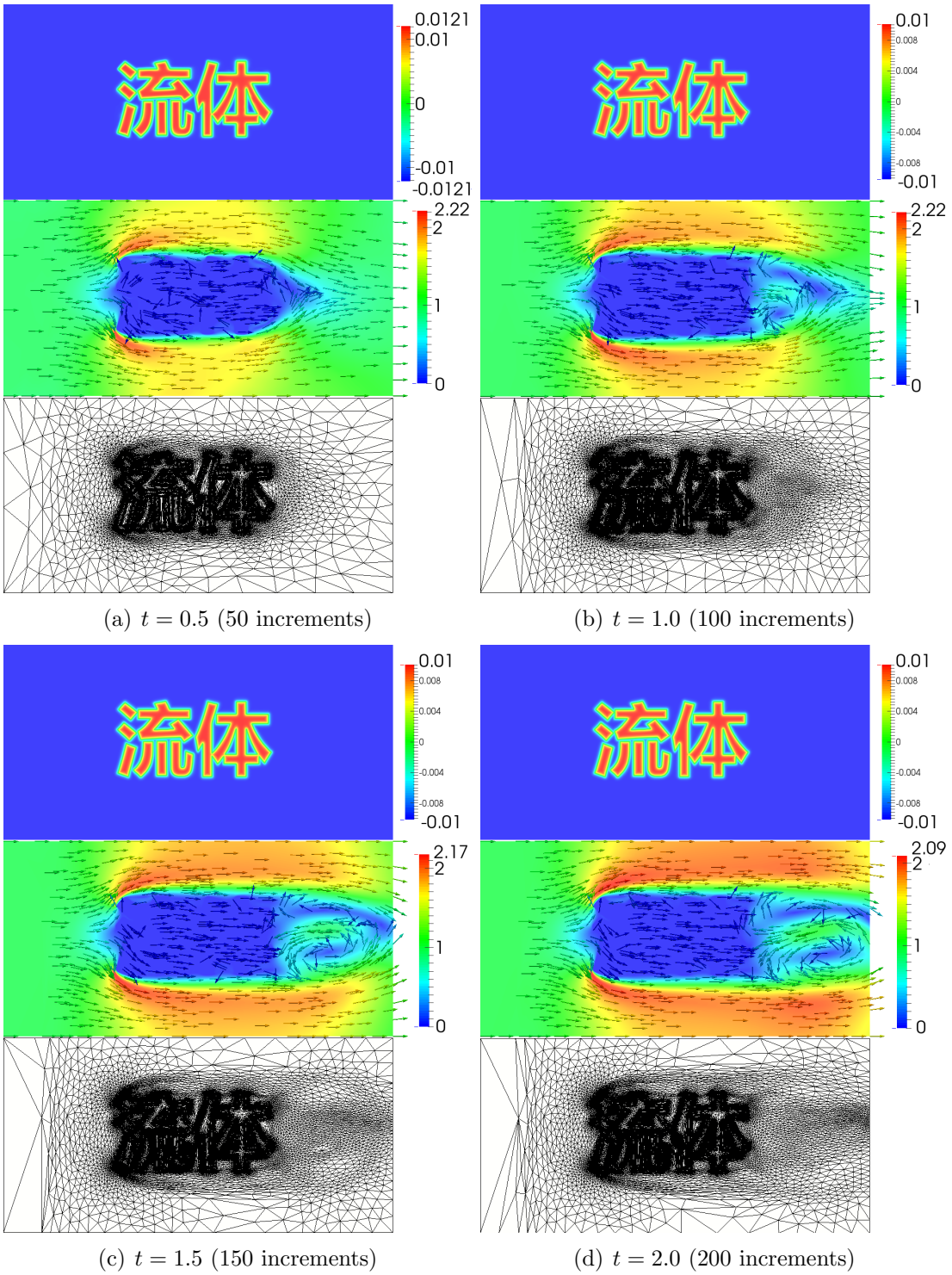


(a) Image drawn with Paint (600×300) (b) Photography of a hand-made drawing (600×300) (c) Segmented image from photography (600×300)

Figure 4.23: Images used on flow simulations directly from drawn images and photography.

In the flow simulation, we have imposed the same boundary conditions as the one considered previously for the flow past cylinder simulations. Numerical parameters chosen are: $\varepsilon = 0.01$; $\Delta\tau = 0.00025$; the number of nodes is fixed at $N = 10000$; the physical parameters of each phase ρ_{solid}, η_{solid} and ρ_{air} η_{air}. Both images have been interpolated in the initial mesh, of dimensions $[0, 2] \times [0, 1]$. Redistancing-Multiphase Navier-Stokes solver and mesh adaptation have been launched, to obtain the redistanced phase function, u_ε^τ, the velocity field, $\mathbf{v}$, and the mesh, at each increment. The initial function u_ε^0 is given from a discontinuous Heaviside function and the redistancing procedure converges after 60 increments. For $Re = 300$ with a flow time step of $\Delta t = 0.01$, the computed velocity field $\mathbf{v}$ and the adapted anisotropic mesh are also illustrated in Figures 4.24 and 4.25.

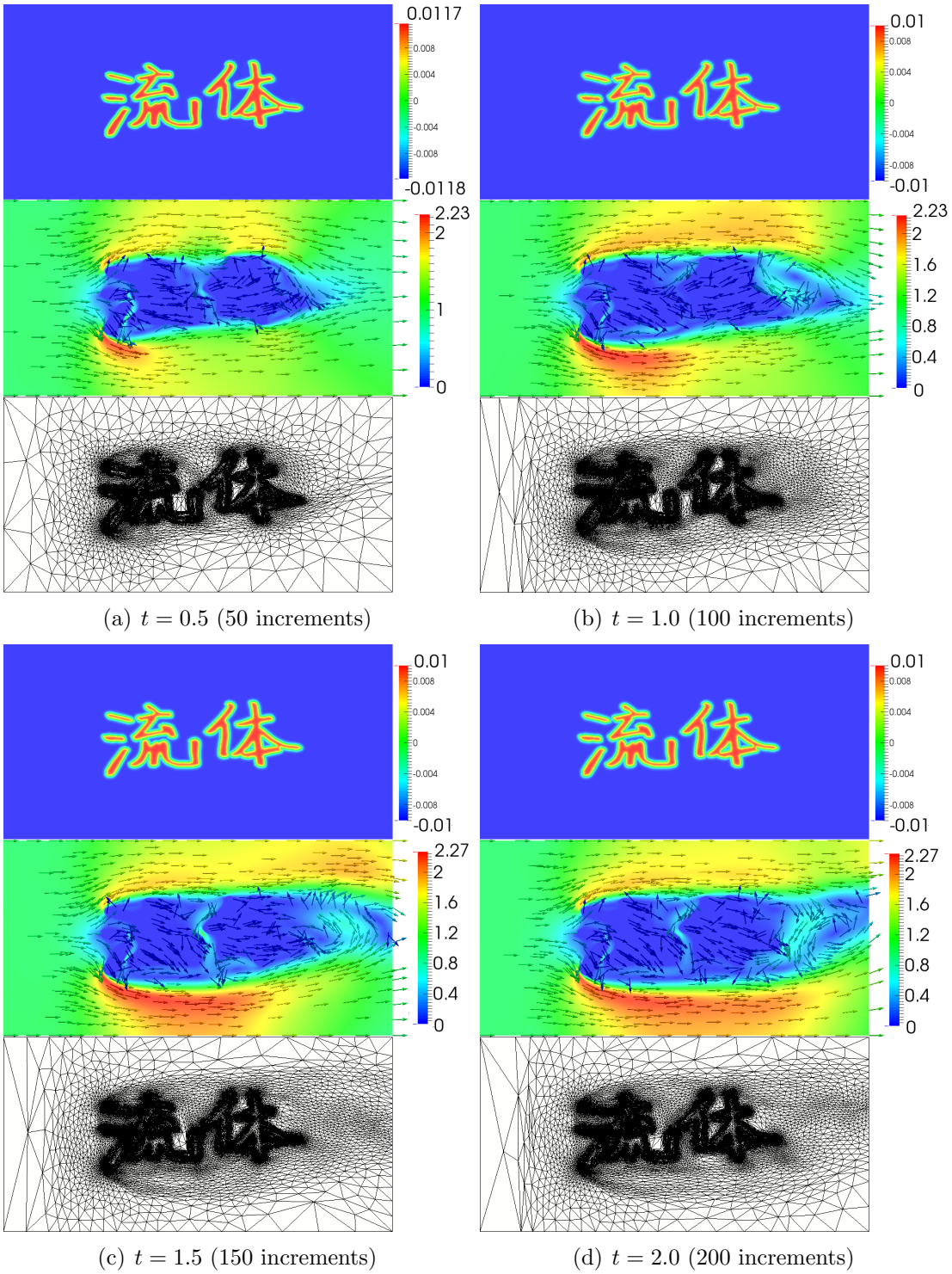
Results obtained, even if qualitative, show an easy way for numerical simulation through our proposed methodology. Paint image guarantees the accuracy and hand-made with photo capture gives a certain flexibility.



(a) $t = 0.5$ (50 increments) (b) $t = 1.0$ (100 increments)

(c) $t = 1.5$ (150 increments) (d) $t = 2.0$ (200 increments)

Figure 4.24: Redistanced function, computed velocity field and adapted anisotropic mesh, for Re $= 300$ at $t = 0.5, 1, 1.5, 2s$.



(a) $t = 0.5$ (50 increments)

(b) $t = 1.0$ (100 increments)

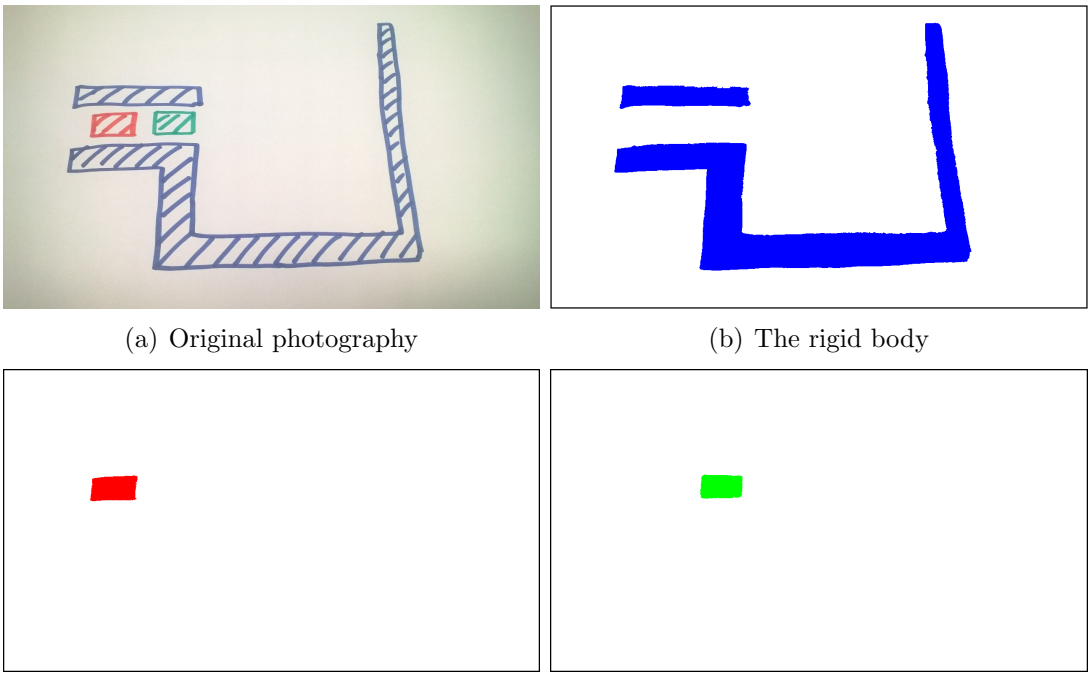
(c) $t = 1.5$ (150 increments)

(d) $t = 2.0$ (200 increments)

Figure 4.25: Redistanced function, computed velocity field and adapted anisotropic mesh, for Re $= 300$ at $t = 0.5, 1, 1.5, 2s$.

4.7.3.2 Moving interfaces and flow simulation

In this section, flow involving moving interfaces, previously presented, will now be done using a photo to describe the geometry. Figure 4.26(a) represents a drawn and after photographed image, of dimensions (1270×720). Three identified colors are visualized: blue, red and green. The blue color represents a cavity rigid region, the red one is the fluid's inlet and the green is the initial fluid filling already the cavity. Here, we propose to simulate a fluid entering the cavity. Flow inlet controls the injection flow rate and we observe the motion of the green part, representing the fluid.



(a) Original photography (b) The rigid body

(c) Flow inlet (d) Initial filling part

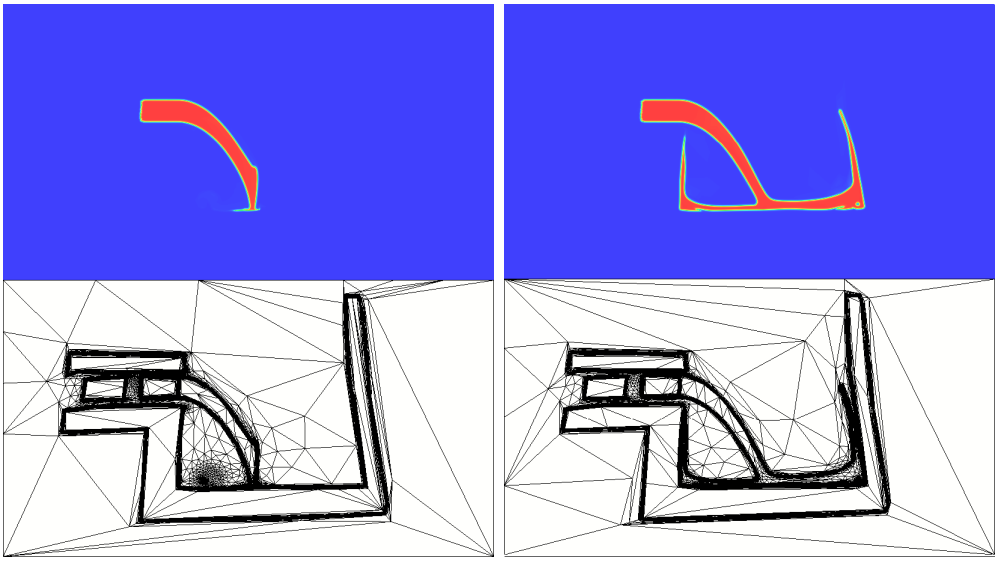
Figure 4.26: Photographed image and its three segmented colors, generating three treated images.

The first step is to segment the three colors using Opening/Closing operations through Morph-M. The results obtained after this processing step are shown in Figures 4.26(b),(c) and (d), where one observes a very good distinction between the different regions.

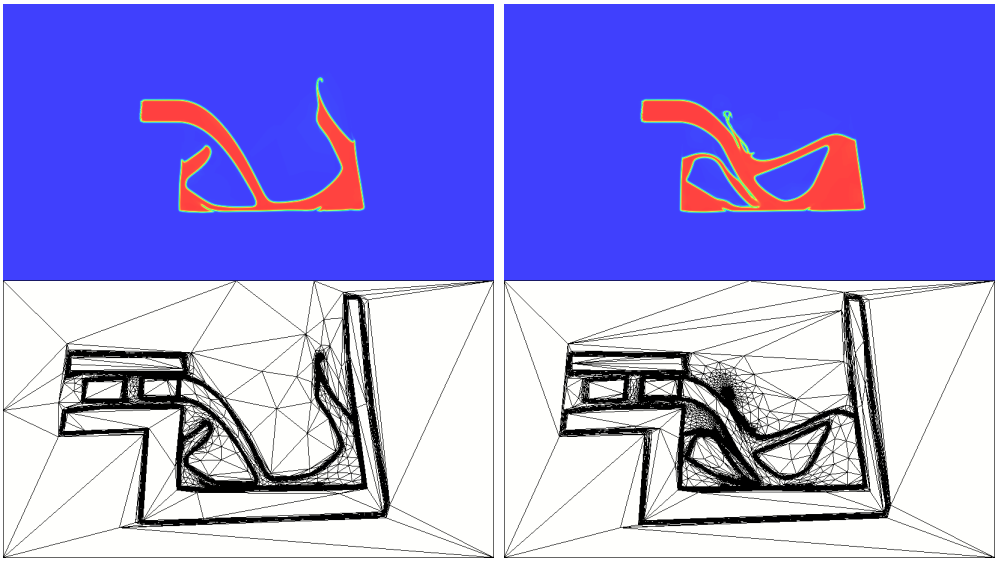
The physical parameters of air, liquid and rigid parts are $\rho_{air} = 1kg/m^3$, $\rho_{fluid} = 1000kg/m^3$, $\rho_{solid} = 20000kg/m^3$ and $\eta_{air} = 0.00002kg/(s{\cdot}m)$, $\eta_{fluid} = 0.1kg/(s{\cdot}m)$, $\eta_{solid} = 1000kg/(s \cdot m)$. The three segmented images are interpolated on an initial mesh of dimensions ($[0, 12.69] \times [0, 7.19]$). The thickness and the fictitious time step are $\varepsilon = 0.01$ and $\Delta\tau = 0.00025$, respectively, and are used for redistancing and for the convected moving interface coupled procedure. The algorithm steps are the same as previously presented, where we have added an initial redistancing procedure.

Let this simulation run on 1 core during 528 minutes (for 5000 increments) with the time step $\Delta t = 0.02$. Figure 4.27 shows the results obtained at $T =$

$25, 50, 75, 100s$, namely the moving convected filling fluid and the adapted anisotropic mesh.



(a) $T = 25s$ (1250 increments) (b) $T = 50s$ (2500 increments)



(c) $T = 75s$ (3750 increments) (d) $T = 100s$ (5000 increments)

Figure 4.27: The convected filling fluid and the adapted anisotropic mesh, at $T = 25, 50, 75, 100s$.

It illustrates the numerical capture of the filling fluid, where the initial and the boundary conditions are given by drawings and photographs. Quality of the drawn picture and camera may influence the results, but overall results are rather interesting, since we manage to define the geometry features in a very simple and easy way.

4.8 Conclusion

In this chapter, a multiphase Navier-Stokes solver has been introduced. It is based on a resolution using a Variational MultiScale finite element method. Mixture laws allow the definition of the physical properties everywhere in the computational domain, using our modified level-set function. This latter can be advected and redistanced simultaneously. Two test cases illustrated the proposed methodology. Furthermore, this methodology based on images that may be issued from different acquisition systems, 2D images or 3D ones, drawn with Paint or hand-made and photographed. All these examples demonstrate the high ability of our methodology to handle numerical simulation on real data, in an accurate, efficient and easy way.

4.9 Résumé français

Dans ce chapitre, nous avons présenté un solveur pour les équations de Navier Stokes multiphasiques. Ce solveur utilise une méthode multi-échelles. Les lois de mélange basées sur la fonction Heaviside régularisée, permettent la définition des propriétés physiques dans tout le domaine de calcul. Cette dernière peut être advecté et réinitializé simultanment. Deux cas test ont illustrée cette méthodologie. Les images utilisées proviennent de différents systémes d'acquisition (photographie, image 2D ou 3D, dessinées à la main ...). Ces exemples montrent la robustesse de la méthode et la facilité de réaliser des simulations numériques de manière simple et précise.

Chapter 5

Conclusions and Perspectives

The main objective of this work aims at the simplification of numerical simulations, by not using CAD software to create numerical geometries, or not creating anisotropic meshes from surface definitions. The only requirement is an "image", no matter in 2D or 3D, grey-scale or color, ... Then, we perform the numerical simulations directly on these image data.

In this work, we have first shown mathematical morphology techniques to process images, providing the segmented regions and distinguishing the phases necessary to our computations. These techniques are efficient and accurate. Then, the immersed image method was proposed to build a "bridge" between the image and the mesh, transforming image data to mesh discretizations.

Then, computing the estimation error for the mesh discretization, the metric tensor field is constructed, which may be used to create the optimal anisotropic mesh. Additionally, we have proposed a new methodology to build a regularized function using the redistancing function. The main advantages are: (1) the rebuild regularized function will be used later for numerical simulations; (2) the gradient of the regularized function is helpful for the anisotropic mesh construction.

Finally, the constructed anisotropic mesh from image data and the redistanced function were directly used in numerical simulations, by coupling it to stabilized finite element solvers. The variation multiscale multiphase Navier-stokes solver was implemented and the illustrated examples have demonstrated how efficient is this methodology. The immersed image method, anisotropic mesh adaptation and the finite element solver are all fully parallel, reducing the computation time, but with a small increase on the memory requirement.

Consequently, this work showed applications of flow computations on MRI scans, $X\text{-}ray$ microtomographies, Smartphone photos and drawings and has proved to be an efficient and powerful methodology in the areas covered by these acquisition systems.

Perspectives of this work concern both the numerical techniques and their applications.

In what concerns numerical developments, further work should be done on adaptation on images. Firstly, extension to 4D images, like films, should be considered. In particular, it would allow an efficient image storage of the space-time image information. Furthermore, the fact that the mesh format may be a way of performing

compression and superpixelisation must be studied in a deeper way and is a very important subject. In whole resolution loop, solvers and adaptation are linked to an error, fixed by the user and different according to the resolution. This error should arise from a global adaptative model, by specifying only the CPU power and/or time. This is something to be pursued in a near future. Tests cases for flow situations were mostly illustrative but on-going work validates the developments here presented.

In terms of applications, efficient image based modelling opens wide perspectives, for all the reasons cited above. The code developed is already used in the medical, material science and urban domains. Comparisons with in-situ experiments should also further validate our approach.

Bibliography

[1] M. Viceconti, C. Zannoni, L. Pierotti, Tri2solid: an application of reverse engineering methods to the creation of cad models of bone segments, Computer Methods and Programs in Biomedicine 56 (1998) 211–220.

[2] W. Lorensen, H. Cline, Marching cubes: A high resolution 3d surface construction algorithm, ACM SIGGRAPH Computer Graphic 21 (1987) 163–169.

[3] D. Rajon, W. Bolch, Marching cube algorithm: review and trilinear interpolation adaptation for image-based dosimetric models, Computational Methods in Image Graphics 27 (2003) 411–435.

[4] The Lenna story (1972).
URL `http://www.lenna.org`

[5] H. Barrow, J. Tenembaum, Interpreting line drawings as three-dimensional surfaces, Artificial Intelligence 17 (1981) 75–116.

[6] T. Lindeberg, Edge detection and ridge detection with automatic scale selection, International Journal of Computer Vision 30 (1998) 117–154.

[7] D. Ziou, S. Tabonne, Edge detection techniques: an overview, International Journal of Pattern Recognition and Image Analysis 8 (1998) 537–559.

[8] D. Ziou, S. Tabonne, Region growing: Childhood and adolescence, Computer Graphics and Image Processing 5 (1976) 382–399.

[9] M. Kass, A. Witkin, D. Terzopoulos, Snakes: active contours model., International Journal of Computer Vision 1 (1988) 321–331.

[10] D. Mumford, J. Shah, Optimal approximation by piece-smooth functions and associated variational problems, Communications on Pure and Applied Mathematics 42 (5) (1989) 577–685.

[11] D. Chan, J. Vese, Active contours without edges., IEEE Transactions on Image Processing 10 (2001) 266–277.

[12] S. Osher, J. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations, Journal of Computational Physics 12-49 (1988) 146–159.

[13] L. Vese, T. Chan, A multiphase level set framework for image segmentation using the mumford and shah model., International Journal of Computer Vision 50 (2002) 271–293.

[14] S. Winkler, C. van den Branden Lambrecht, M. Kunt, Vision models and applications to image and video processing, Springer, 2001, Ch. Vision and Video: Models and Applications, p. 209.

[15] N. Ahmed, T. Natarajan, K. Rao, Discrete cosine transform, IEEE Transactions on Computers C-23 (1974) 90–93.

[16] Official joint photographic experts group site (1994).
URL `http://www.jpeg.org`

[17] A. Jacquin, A fractal theory of iterated markov operators with applications to digital image coding, Ph.D. thesis, Georgia Institute of Technology (1989).

[18] J. Murray, W. vanRyper, Encyclopedia of Graphics File Formats, 2nd Edition, O'Reilly Media, 1996.

[19] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Transactions on Information Theory 24 (5) (1978) 530.

[20] T. Welch, A technique for high-performance data compression, Computer 17 (6) (1984) 8–19.

[21] B. Delaunay, Sur la sphere vide, Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7 (793-800) (1934) 1–2.

[22] W. Frey, Selective refinement: A new strategy for automatic node placement in graded triangular meshes, International Journal for Numerical Methods in Engineering 24 (11) (1987) 2183–2200.

[23] F. Hecht, Bidimensional anisotropic mesh generator, technical report, inria, rocquencourt.
URL `http://www.ann.jussieu.fr/hecht/ftp/bamg/bamg.pdf`

[24] M. Yerri, M. Shephard, Automatic 3d mesh generation by the modified-octree technique, International Journal for numerical methods in engineering 20 (1984) 1965–1990.

[25] B. Chaudhuri, Applications of quadtree, octree, and binary tree decomposition techniques to shape analysis and patern recognition, IEEE Transactions on pattern analysis and machine intelligence PAMI-7 (6) (1985) 652–661.

[26] P. George, E.Seveno, The advancing-front mesh genenration method revisited, International Journal for numerical methods in engineering 37 (1994) 3605–3619.

[27] T. Coupez, Grandes transformations et remaillage automatique, Ph.D. thesis, Ecole Nationale Superieure des Mines de Paris, CEMEF, Sophia Antipolis,France (1991).

[28] T. Coupez, Génération de maillage et adaptation de maillage par optimisation locale, Revue Europeenne d'Elements Finis 9 (4) (2000) 403–423.

[29] C. Gruau, T. Coupez, 3d tetrahedral unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4951–4976.

[30] I. Babuška, W. Rheinboldt, A-posteriori error estimates for the finite element method., International Journal for Numerical Methods in Engineering 12 (10) (1978) 1597–1615.

[31] C. Zinekiewicz, J. Zhu, The supercovergent patch recovery and a posteriori estimates. part 1: the recovery technique, International Journal for Numerical Methods in Engineering 33 (1992) 1331–1364.

[32] T. Brooks, A. Hughes, Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations, Computer Methods in Applied Mechanics and Engineering 32 (1) (1982) 199–259.

[33] T. Hughes, M. Mallet, A new finite element formulation for computational fluid dynamics: Iii. the generalized streamline operator for multidimensional advective-diffusive systems, Computer Methods in Applied Mechanics and Engineering 58 (3) (1986) 305–328.

[34] F. Brezzi, M. Bristeau, L. Franca, M. Mallet, G. Rogé, A relationship between stabilized finite element methods and the galerkin method with bubble functions, Computer Methods in Applied Mechanics and Engineering 96 (1) (1992) 117–129.

[35] F. Brezzi, A. Russo, Choosing bubbles for advection-diffusion problems, Mathematical Models and Methods in Applied Sciences 4 (4) (1994) 571–587.

[36] T. Hughes, G. Feijoo, L. Mazzei, J. Quincy, The variational multiscale method a paradigm for computational mechanics, Computer Methods in Applied Mechanics and Engineering 166 (1998) 3–24.

[37] T. Coupez, Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing, Journal of Computational Physics 230 (2011) 2391–2405.

[38] T. Coupez, G. Jannoun, N. Nassif, H. Nguyen, H. Digonnet, E. Hachem, Adaptive time-step with anisotropic meshing for incompressible flows, Journal of Computational Physics 241 (2013) 195–211.

[39] Brainweb: Simulated brain database.
URL `http://www.bic.mni.mcgill.ca/brainweb`

[40] B. Aubert-Broche, M. Griffin, G. Pike, A. Evans, D. Collins, Twenty new digital brain phantoms for creation of validation image data bases, IEEE Transactions on Medical Imaging 25 (11).

[41] B. Aubert-Broche, A. Evans, D. Collins, A new improved version of the realistic digital brain phantom, NeuroImage 32 (138-145).

[42] P. Frey, F. Alauzet, Anisotropic mesh adaptation for cfd computations, Computer Methods in Applied Mechanics and Engineering 194 (2005) 5068–5082.

[43] O. Basset, Simulation numérique découlements multi-fluides sur grille de calcul. mechanics, Ph.D. thesis, Ecole Nationale Superieure des Mines de Paris,France (2006).

[44] F. Alauzet, Size gradation control of anisotropic meshes, Finite Element in Analysis and Design 46 (2010) 181–202.

[45] H. Huang, Metric tensors for anisotropic mesh generation, Journal of Computational Physics 204 (2005) 633–665.

[46] F. Alauzet, Adaptation de maillage anisotrope en trois dimensions. application aux simulations instationnaires en mécanique des fluides, Ph.D. thesis, Universite Montpellier II, Montpellier, France (2003).

[47] A. Loseille, Adaptation de maillage anisotrope 3d multi-échelles et ciblé á une fonctionnelle pour la mécanique des fluides. application á la prédiction haute-fidélité du bang sonique, Ph.D. thesis, Université Pierre et Marie Curie, Paris VI, Paris, France (2003).

[48] T. Coupez, A mesh improvement method for 3d automatic remeshing, numerical grid generation in computational fluid dynamics and related fields, Pineridge Press (1994.) 615–626.

[49] Ge healthcare: Optima mr450w 1.5t with gem suite.
URL `http://www3.gehealthcare.com`

[50] T. Coupez, H. Digonnet, R. Ducloux, Parallel meshing and remeshing, Applied Mathematical Modelling 25 (2000) 83–98.

[51] L. Shapiro, G. Stockman, Computer Vision, Prentice Hall, 2001.

[52] C. Li, C. Kao, J. Gore, Z. Ding, Implicit active contours driven by local binary fitting energy, in: IEEE: Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–7.

[53] T. Coupez, E. Hachem, Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing, Computer Method in Applied Mechanics and Engineering (2013) 65–85.

[54] E. Hachem, H. Digonnet, E. Massoni, T. Coupez, Immersed volume method for solving natural convection, conduction and radiation of a hat-shaped disk inside a 3d enclosure, International Journal of Numerical Methods for Heat and Fluid Flow 22 (6) (2012) 2–2.

[55] E. Hachem, T. Kloczko, H. Digonnet, T. Coupez, Stabilized finite element solution to handle complex heat and fluid flows in industrial furnaces using the immersed volume method, International Journal for Numerical Methods in Fluids 68 (1) (2012) 99–121.

[56] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1) (1994) 146–159.

[57] M. Sussman, E. Fatemi, An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM Journal on Scientific Computing 20 (4) (1999) 1165–1191.

[58] T. Coupez, L. Silva, E. Hachem, Implicit boundary and adaptive anisotropic meshing, EMA SIMAI Springer Series.

[59] L. Franca, S. Frey, T. Hughes, Stabilized finite element methods: I. application to the advective-diffusive model, Computer Methods in Applied Mechanics and Engineering 95 (2) (1992) 253–276.

[60] R. Codina, Comparison of some finite element methods for solving the diffusion-convection-reaction equation, Computer Methods in Applied Mechanics and Engineering 156 (1-4) (1998) 185–210.

[61] E. Hachem, Stabilized finite element method for heat transfer and turbulent ows inside industrial furnaces, Ph.D. thesis, Ecole Nationale Superieure des Mines de Paris, CEMEF, Sophia Antipolis,France (2009).

[62] L. Franca, A. Russo, Deriving upwinding, mass lumping and selective reduced integration by residual-free bubbles, Applied Mathematics Letters 9 (5) (1996) 83–88.

[63] A. Masud, R. A. Khurram, A multiscale/stabilized finite element method for advection-diffusion equation, Computer Methods in Applied Mechanics and Engineering 193 (2004) 1997–2018.

[64] Morph-m: Image processing library specialized in mathematical morphology.
URL `http://cmm.ensmp.fr/Morph-M/`

[65] Online course on mathematical morphology.
URL `http://cmm.ensmp.fr/ serra/cours/index.htm`

[66] Centre for mathematical morphology.
URL `http://cmm.ensmp.fr`

[67] J. Serra, Image Analysis and Mathematical Morphology, Academic Press, London, 1982, Ch. Volume 1, p. 600.

[68] J. Serra, Image Analysis and Mathematical Morphology, Academic Press, London, 1988, Ch. Volume 2: Theoretical Advances, p. 411.

[69] S. Beucher, Mathematical Morphology and its Applications to Image Processing, Kluwer Academic Publishers, Fontainebleau, 1994, Ch. Watershed, hierarchical segmentation and waterfall algorithm, pp. 69–76.

[70] L. Orgéas, P. Dumont, J. Vassal, O. Guiraud, V. Michaud, D. Favier, In-plane conduction of polymer composite plates reinforced with architectured networks of copper fibres, Journal of Materials Science 47 (2932-2942).

[71] S. E. Feghali, Novel monolithic stabilized finite element method for fluid-structure interaction, Ph.D. thesis, Ecole Nationale Superieure des Mines de Paris, CEMEF, Sophia Antipolis,France (2012).

[72] J. Donea, S. Giuliani, J. P. Halleux, An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions, Computer Methods in Applied Mechanics and Engineering 33 (1982) 689–723.

[73] W. A. Wall, S. Genkinger, E. Ramm, A strong coupling partitioned approach for fluidstructure interaction with free surfaces, Computers and Fluids 36 (2007) 169–183.

[74] B. Hubner, E. Walhorn, D. Dinkler, A monolithic approach to fluid-structure interaction using space-time finite elements, Computer Methods in Applied Mechanics and Engineering 193 (23-26) (2004.) 2087–2104.

[75] E. Walhorn, A. Kolke, B. Hubner, D. Dinkler, Fluid-structure coupling within a monolithic model involving free surface flows, Computers and structures 83 (2005) 2100–2111.

[76] R. Codina, Stabilized finite element approximation of transient incompressible flows using orthogonal subscales, Computer Methods in Applied Mechanics and Engineering 191 (2002) 4295–4321.

[77] R. Codina, Finite element approximation of the three field formulation of the stokes problem using arbitrary interpolations, SIAM Journal on Numerical Analysis 47 (2009) 699–718.

[78] S. Badia, R. Codina, Stabilized continuous and discontinuous galerkin techniques for darcy flow, Computer Methods in Applied Mechanics and Engineering 199 (2010) 1654–1667.

[79] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, Journal of Computational Physics 183 (2002) 83–116.

[80] J. Bruchon, T. Coupez, Etude de la formation dune structure de mousse polymere par simulation de lexpansion anisotherme de bulles de gaz, Mecanique et Industries 4 (2003) 331.

[81] D. Peng, B. Merriman, S. Osher, H. Zhao, , M. Kang, A pde-based fast local level set method, Journal of Computational Physics 155 (1999) 410–438.

[82] L. Ville, L. Silva, T. Coupez, Convected level set method for the numerical simulation of fluid buckling, International Journal for Numerical Methods in Fluids 66 (3) (2011) 324–344.

[83] L. Ville, Modélisation multiphasique et calcul d'interface dans les procédés de mise en oeuvre des propergols, Ph.D. thesis, Ecole Nationale Superieure des Mines de Paris, CEMEF, Sophia Antipolis,France (2011).

[84] A. Placzek, J. Sigrist, A. Hamdouni, Numerical simulation of an oscillating cylinder in a cross-flow at low reynolds number: Forced and free oscillations, Computers and Fluids 38 (2009) 80–100.

[85] M. Gerouache, Etude numérique de l'instabilité de bénard-kármán derrière un cylindre fixe ou en mouvement périodique. dynamique de l'écoulement et advection chaotique., Ph.D. thesis, Ecole Polytechnique de l'Université de Nantes,France (2000).

[86] C. Norberg, Fluctuating lift on a circular cylinder: review and new measurements., Journal of Fluids and Structures 17 (1) (2003) 57–96.

Résumé

Ces dernières années, les techniques d'imagerie ont fait l'objet de beaucoup d'améliorations. Elles permettent de fournir des images numériques 2D ou 3D précises de zones parfois invisibles à l'oeil nu. Dans cette thèse, l'imagerie sera utilisée pour effectuer des simulations numériques en la couplant avec un solveur éléments finis. Nous présenterons, en premier lieu, la morphologie mathématique et la méthode d'immersion d'image. Elles permettront l'extraction d'informations permettant la transformation d'une image dans un maillage exploitable. Puis, une méthode itérative d'adaptation de maillage basée sur un estimateur d'erreur sera utilisée afin de construire un maillage optimal. Ainsi, un maillage sera construit uniquement avec les données d'une image. Nous proposerons également une nouvelle méthodologie pour construire une fonction régulière à l'aide d'une méthode de réinitialisation de la distance signée. Deux avantages sont à noter : l'utilisation de la fonction régularisée permet une bonne adaptation de maillage. De plus, elle est directement utilisable par le solveur éléments finis. Les simulations numériques sont donc réalisées en couplant éléments finis stabilisés, adaptation de maillage anisotrope et réinitialisation. L'objectif de cette thèse est donc de simplifier le calcul numérique à partir d'image, d'améliorer la précision numérique, la construction d'un maillage automatique et de réaliser des calculs numériques parallèles efficaces. Les applications envisagées peuvent être dans le domaine médical, de la physique des matériaux ou du design industriel.

Mots Clés

Traitement de l'image, la fonction level-set, la fonction de réinitialisation, anisotrope adaptation de maillage, méthode des éléments finis

Abstract

Imaging techniques have well improved in the last decades. They may accurately provide numerical descriptions from 2D or 3D images, opening perspectives towards inner information, not seen otherwise. In this work, a technique to build a numerical description under the mesh format has been implemented and used in numerical simulations when coupled to finite element solvers. Firstly, mathematical morphology techniques have been introduced to handle image information, providing the specific features of interest for the simulation. The immersed image method was then proposed to interpolate the image information on a mesh. Then, an iterative anisotropic mesh adaptation operator was developed to construct the optimal mesh, based on the estimated error concerning the image interpolation. The mesh is thus directly constructed from the image information. We have also proposed a new methodology to build a regularized phase function, corresponding to the objects we wish to distinguish from the image, using a redistancing method. Two main advantages of having such function are: the gradient of the regularized function performs better for mesh adaptation; the regularized function may be directly used for the finite element solver. Stabilized finite element flow and advection solvers were coupled to the constructed anisotropic mesh and the redistancing function, allowing its application to multiphase flow numerical simulations. An important objective of this work is the simplification of the image based computations, through a modified way to segment the image and by coupling all to an automatic way to construct the mesh used in the finite element simulations.

Keywords

Image processing, level-set function, redistancing function, anisotropic mesh adaptation, finite element method